AFHRL-TR-89-25

KEATS: A SYSTEM TO SUPPORT KNOWLEDGE ENGINEERING AND TRAINING FOR DECISION-MAKING SKILLS

Fritz H. Brecke
Patrick Hays
Donald Johnston
Gail Slemon
Jane McGarvey
Susan Peters

Logicon, Incorporated
Tactical and Training Systems Division
4010 Sorrento Valley Boulevard
San Diego, California 92138-5158

LOGISTICS AND HUMAN FACTORS DIVISION
Wright-Patterson Air Force Base, Ohio 45433-6503

December 1989
Interim Technical Report for Period November 1986 - May 1989

AIR FORCE HUMAN RESOURCES LABORATORY

AIR FORCE SYSTEMS COMMAND
BROOKS AIR FORCE BASE, TEXAS 78235-5601

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

The Public Affairs Office has reviewed this report, and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nationals.

This report has been reviewed and is approved for publication.

MICHAEL J. YOUNG, Contract Monitor
Logistics and Human Factors Division

BERTRAM W. CREAM, Technical Director
Logistics and Human Factors Division

HAROLD G. JENSEN, Colonel, USAF
Commander

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1989	Interim -- November 1986 to May 1989

4. TITLE AND SUBTITLE

KEATS: A System to Support Knowledge Engineering and Training for Decision-Making Skills

5. FUNDING NUMBERS

C - F33615-86-C-0020
PE - 62205F
PR - 3017
TA - 08
WU - 15

6. AUTHOR(S)

Fritz H. Brecke, Donald Johnston, Jane McGarvey
Patrick Hays, Gail Slemon, Susan Peters

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Logicon, Incorporated
Tactical and Training Systems Division
4010 Sorrento Valley Boulevard
San Diego, California 92138-5158

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Logistics and Human Factors Division
Air Force Human Resources Laboratory
Wright-Patterson Air Force Base, Ohio 45433-6503

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

AFHRL-TR-89-25

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report presents the development of advanced training technologies for decision-making skills. Training systems must, like decision aids, be based on expert knowledge of the target decision domain. Initial knowledge acquisition activities using structured and focused interview techniques to access the domain knowledge of decision makers in Air Force command and control nodes (Air Support Operations Centers or ASOCs) led to the identification of five types of methodological problems. The report describes and illustrates these problems, as well as a computer-based tool which is at least a partial solution to three of the five problems. This tool, the Knowledge Engineering and Training System (KEATS), is a software shell written in Smalltalk which runs on PC/AT-type machines. KEATS permits the utilization of domain expert knowledge in the generation of detailed and realistic decision scenarios. Once created, these decision scenarios provide the stimulus for either training or knowledge engineering. KEATS was subjected to a 2-day formative evaluation trial with two experts. The trial was a successful demonstration of the concept of computer-aided knowledge engineering. The results are described, and conclusions and recommendations are offered.

Keywords:

14. SUBJECT TERMS

computer-based training
decision-making training
expert knowledge
knowledge acquisition
knowledge engineering
microcomputers
tactical command and control

15. NUMBER OF PAGES

18

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

SUMMARY

Battle managers assigned to Air Force Tactical Air Control Systems need to be competent decision makers. Improved technologies are required to develop and deliver training in decision-making skills at an affordable cost. Training should be available at peacetime duty stations, run on microcomputers available at the squadron level, and not require instructors.

The initial focus of the present effort was on acquisition of instructional content. Instructional content for decision training consists of the task knowledge employed by experts during the act of making decisions. This task knowledge resides in the minds of experts and must be made explicit. One way is with the aid of knowledge engineering techniques. Task analysis techniques have proven to be ineffective for this purpose. Unaided, interview-type knowledge engineering methods were successful up to a point but encountered significant problems. A computer-based (Smalltalk and PC/AT) Knowledge Engineering and Training System (KEATS) was developed to aid in the knowledge acquisition process and to overcome the problems encountered. KEATS is the first of two training system prototypes to be developed under this project.

To date, KEATS has been used as a knowledge engineering support tool during a 2-day formative evaluation trial with two experts. Schemas for situation assessment and plans, and 64 rules for air request tasking, were identified, along with 23 requirements for functional changes of KEATS.

Although much additional work remains to be done, the results are interpreted as an initial proof of concept for the KEATS tool. More empirical tests are required to determine whether the problems that gave rise to KEATS development are solved. Continued use of KEATS for knowledge acquisition to develop a second training system and use of KEATS in similar decision domains is recommended.

PREFACE

With its establishment in 1980, the Logistics and Human Factors Division's Ground Operations Branch, a part of the Air Force Human Resources Laboratory (AFHRL/LRG) at Wright-Patterson AFB, began a comprehensive program to develop advanced techology to improve training for USAF Tactical Command and Control (C^2) Battle Staff personnel.

This report covers a research project which centered around the use of knowledge engineering techniques to acquire instructional content for tactical C^2 decision training. A Knowledge Engineering and Training System (KEATS) was developed. This document describes why KEATS was developed, its functional characteristics, and its first application.

The project is guided by Mr. Michael J. Young of AFHRL/LRG and executed by Logicon's Tactical and Training Systems Division in San Diego, California, under the direction of Fritz H. Brecke. Logicon team members include: Patrick Hays, Donald Johnston, Gail Slemon, Jane McGarvey, and Susan Peters. Particularly heartfelt thanks go to the Air Force subject-matter experts: Lt Col Fred Wilson, Lt Col Lynn Weber, Lt Col Matt Szczepanek, and Maj Ken Dekay.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. PROBLEM

Effective utilization of tactical combat assets depends to a very large extent on the cognitive skills of personnel who assign combat assets to objectives or targets. Such decision-making functions are generally performed by "battle managers"; i.e., officers assigned to Tactical Command and Control (C^2) systems in all services. Rapid, accurate decision making is therefore a critical combat skill for tactical battlestaff officers. This skill is expected to be acquired during various types of large- and small-scale exercises which occupy most of the peacetime duty time (over 80%) of officers assigned to the Air Force Tactical Air Control System (TACS). Small-scale exercises, which are usually controlled and staged by individual nodes of the TACS, focus on procedural and mechanical skills. Large-scale exercises are usually the only training opportunities for complex tactical C^2 decision-making skills.

Large-scale exercises can involve many TACS nodes or an entire TACS network. They are of two types: Field Training Exercises (FTXs), which involve actual troop and aircraft movements; or Command Post Exercises (CPXs), which involve simulated movements. The best currently available technology for decision-making training is that used in CPXs; i.e., computer-based simulations running on medium- and large-scale computer systems. Simply buying more of this technology and thus solving the problem of training availability would be a prohibitively expensive solution to the decision training problem. There is a clear need for technology that utilizes desk-top microcomputers.

Developing such technology means more than merely offloading current tactical battle simulation technologies to microcomputers or, vice versa, simply waiting for microcomputer technology to reach the required levels of computing power and storage capacity. Current simulation technology is still manpower-intensive and thus impractical at the unit level. Means must therefore be found to replace human participants with intelligent artificial participants and to reduce exercise preparation costs by well-designed exercise editing packages. And finally, means must be found to model a reasonably intelligent and hostile opponent who follows doctrines and procedures that are "alien" (i.e., different from ours). All of this is becoming increasingly feasible; the modeling problem is becoming easier to solve with modern programming techniques and languages, and hardware that is powerful enough to run such models is now available.

Developing decision training technology is ultimately not a task that can be accomplished by computer engineering alone; it takes "cognitive engineering" (Rasmussen, 1986) as well. Montague (1986) expressed this notion as follows: "The analysis of competent performance and its development that must be done to plan instruction must include cognitive organization and structures, and attend to the phases

and processes involved in acquisition." In more general terms, any attempt at developing improved training and/or training design technologies should really be an attempt at translating a theory into common practice.

The current project was launched because there is good reason to believe a relevant scientific base has emerged in recent years as a consequence of progress made in cognitive psychology and cognitive science. Since Nickerson and Feehrer's (1975) landmark report, an entire new body of knowledge has emerged that deals with the nature and acquisition of expertise. This work, much of which was sparked by research and development (R&D) on differences between novices and experts, has prompted a keen appreciation of the role of domain knowledge in expertise and has spawned powerful new concepts and theories dealing with the representation of knowledge and its acquisition. If it is possible to synthesize this new research knowledge to the extent that prescriptive instructional design principles can be formulated, then more elegant solutions to the decision training problem can be put into practice than those represented by the raw high-fidelity simulation approach. Hopefully such solutions can be implemented with affordable and practically available computer resources.

The need for more affordable decision training technology has been recognized and R&D efforts to produce such technologies have been accomplished. Wilson (1982) wrote a master's thesis at the Naval Postgraduate School describing an "Interactive Micro-Computer Wargame for an Air Battle" which ran on an Apple III and was written in UCSD Pascal. Obermayer, Johnston, Slemon, and Hicklin (1984) reported the development of a micro-based (WICAT System 150) multi-user system which simulated a simple Anti-Submarine Warfare (ASW) scenario. The system served as a research prototype; it required player decision inputs and featured one artificial, knowledge-based team member. Madni, Ahlers, and Chu (1987) demonstrated the modeling of an intelligent opponent in a knowledge-based simulation prototype running on a Symbolics 3670 workstation. The prototype was designed to train Tactical Action Officers in the kind of decision-making skills needed during naval surface warfare. The Army Research Institute for the Behavioral and Social Sciences has focused on experimental Computer-Aided Instruction (CAI) in the training of decision-making skills for armor officers. One result of this work is the "Armor Tactical Concepts Tutor (ARTACT)," which runs on the Army's microcomputer-based Electronic Information Delivery System (EIDS). Stoddard, Kern, and Emerson (1986) announced the development of a cognitive skills tutor which continues and expands the ARTACT work.

For the program reported here, the overall goal was to add to this emerging technological base that enables training of combat-critical, complex cognitive skills (such as decision making). The specific objective is a set of training *design* guidelines for creating interactive training experiences which:

Reliably cause the acquisition of decision-making skills; and

Are presented by means of *micro*computer media *without a human instructor*.

This objective is restrictive because we are interested in a method to design training systems for a specific purpose; namely, training of decision-making skills. Ideally, the design methodology should be employable by subject-matter experts (SMEs) without the help of training experts. This ideal may not be attainable within this effort but it is the overall goal.

This report covers work accomplished during the first 2 years of a 3-year effort. The report covers approaches to the overall problem and the subproblems, and results to date. The results are discussed from the perspective of the current stage of the project and much may change as the project goes through its final phase. The report closes with a projection as to what we hope to achieve by the end of the third and final project year.

II. APPROACH

Overview

The approach used in this project is illustrated in Figure 1. The project in its entirety is viewed as an Instructional Systems Design (ISD) problem, where a generic training design solution for a class of skills is sought. The inputs to the problem are the following broadly specified instructional variables:

a.	*Training Objective*	Specified as a class of skills to be trained: Decision-Making Skills
b.	*Target Population*	Battlestaff Officers assigned to elements of Air Force Tactical Command and Control Systems
c.	*Delivery System*	Microcomputer resources available at wing or squadron level; self-instructional methods (no instructor)

The desired output is empirically validated design guidelines that can be used to develop training packages which reliably teach tactical C^2 decision-making skills.

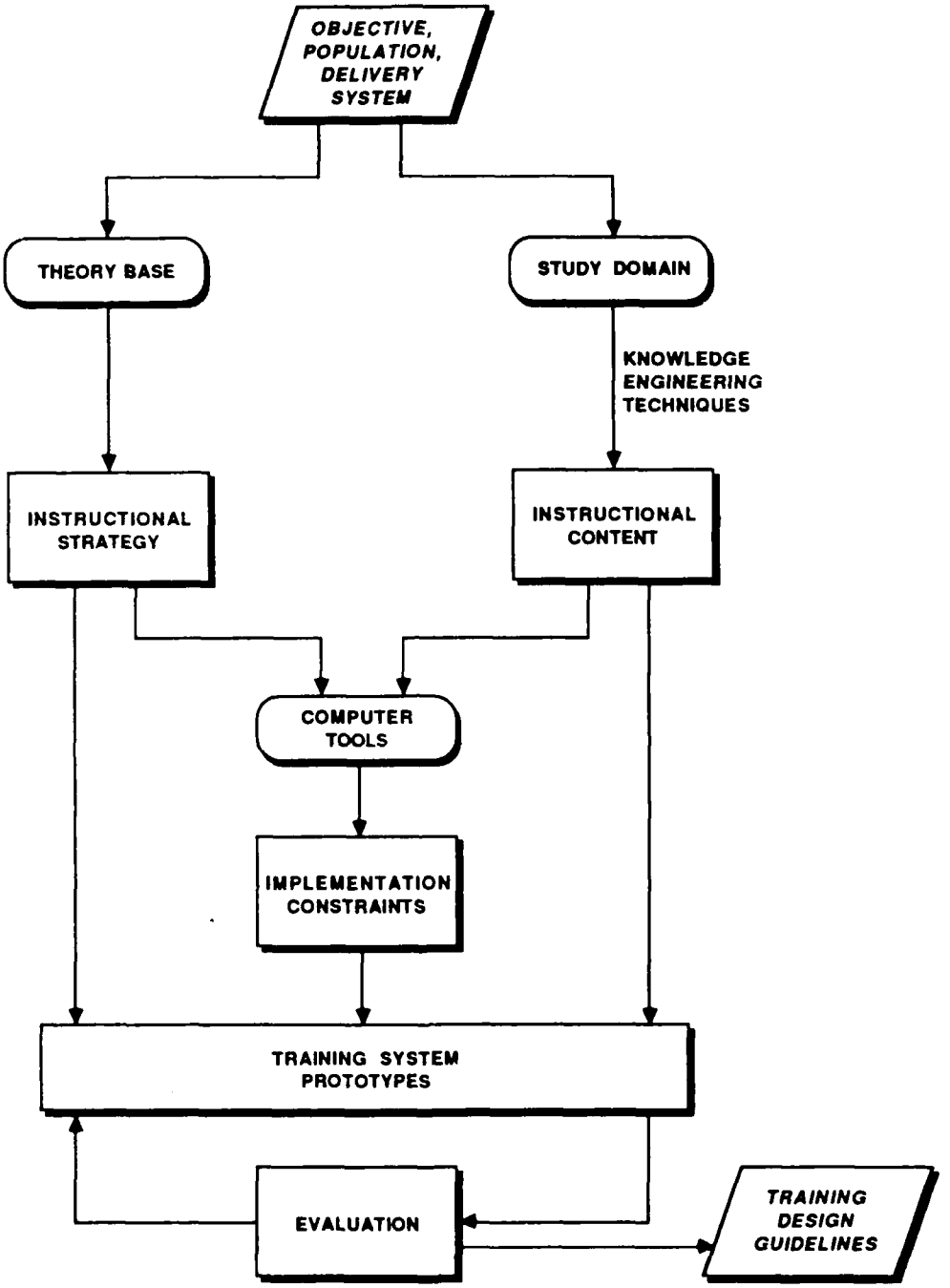

Figure 1. Approach to Training Technology Development.

In order to solve this general ISD problem, two additional instructional variables, besides those specified as problem inputs, must be defined (see Merrill & Wood, 1974; for a more formal treatment, see Frank, 1969):

a.	*Instructional Strategy*	Types of instructional actions and the conditions for their sequencing
b.	*Instructional Content*	Classes and instances of domain knowledge

A generic instructional strategy for training decision-making skills is either available (in ISD manuals) or should be specifiable from a theoretical and empirical base. The instructional content is the domain knowledge employed by battle managers during the performance of decision-making tasks. This knowledge does not exist in explicit form (manuals or regulations).

Making this knowledge explicit is difficult. An earlier effort (Brecke, Jacobs, & Krebs, 1988) showed that conventional ISD task analysis methods work well for procedural skills but are ineffective in capturing the detailed task knowledge employed in higher level cognitive tasks. An alternate technique, developed and employed by the same researchers, approached the problem by eliciting cognitive maps as representations of task knowledge. This technique, while much more efficient as a task analysis method, was equally ineffective in capturing task knowledge for cognitive tasks.

The approach chosen in this project is based on the idea of applying knowledge engineering techniques used in expert systems development to the problem of acquiring cognitive task knowledge or instructional content. This approach requires access to personnel who are expert decision makers in some subdomain of Tactical Command and Control. The subdomain should be representative of the larger domain of Tactical Command and Control to allow later generalization. It should also be small and bounded enough to allow treatment within the project's time and manpower constraints.

Once the two variables of instructional content and strategy are defined, it is possible to specify, design, and develop prototype training systems that are specific instantiations of the generic instructional strategy, that use the acquired instructional content, and that reflect the implementation constraints imposed by the features and limitations of extant microcomputer hardware and software.

The prototypes should be subjected to cycles of formative evaluation trials and revisions. The eventual product is a validated instructional strategy, together with guidelines for its application to particular decision domains. This is what Montague (1986) called "instructional design heuristics."

It should be noted that this approach is idealized, especially with respect to the repeated evaluation and revision cycles of multiple training system prototypes. The funding for this project does not permit more than the development of two limited prototypes and very limited evaluation work. Nevertheless, it provides the necessary beginning.

REPORT SCOPE

Phase I of the project (4 months in duration) was devoted to exploration and preparation. Four major activities were performed:

1. Exploration of the theory base which could give rise to prescriptive instructional design principles or instructional strategies,
2. Identification and exploration of a suitable decision-making domain,
3. Evaluation and selection of hardware and software to be used as a development environment, and
4. Development of concepts and initial architectures for a decision-making training system prototype.

During Phase II, work concentrated on the *acquisition of domain knowledge* and on the search for a generic *instructional strategy*. The latter is in progress but not yet far enough along to justify treatment in the present report. The efforts in domain knowledge acquisition have led to the development of a computer-based knowledge acquisition support system that potentially can double as a training system for decision-making skills. This system is a prototype of a new technology. The remainder of this report describes this technology in detail.

The Study Domain

The general target domain for the project is the Tactical Command and Control System of the Air Force. Within the general domain of Air Force Tactical Command and Control, the focus was on a specific type of Tactical Command and Control node called the Air Support Operations Center (ASOC). The ASOC is an Air Force Tactical Command and Control unit which is "assigned" to and co-located with an Army Corps Headquarters (see Figure 2). Its basic role is to supply air support missions in response to demands originating from the Army Corps' front-line divisions. The ASOC thus has to solve a problem which, in general terms, can be viewed as a supply management problem. The problem is nontrivial because the demand for air support may exceed the supply of available aircraft, and both supply *and* demand vary independently over time in ways that are only partially predictable.

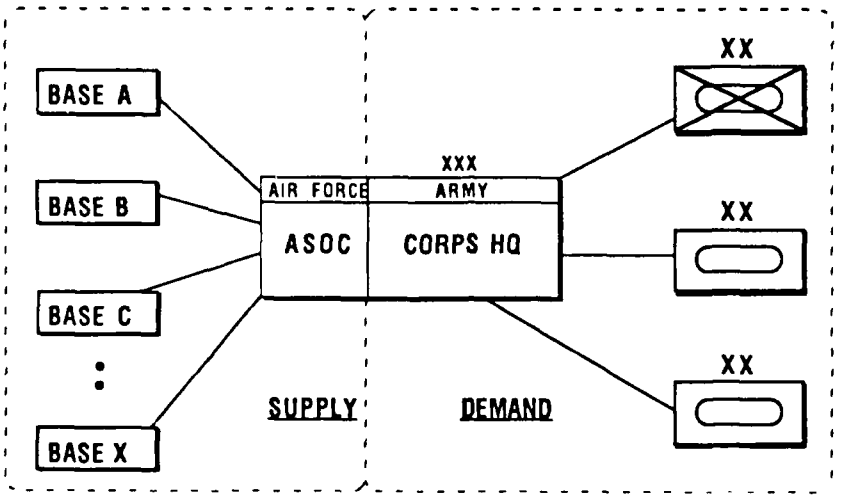

Figure 2. Project Study Domain: Air Support Operations Center (ASOC).

PLANNING AND OPERATIONS AT THE ASOC

The ASOC's method for solving this supply management problem is a combination of advance planning and subsequent plan execution with ad-hoc adjustments. Advance planning is done during the night shift when little or no combat action is occurring.[1] Plan execution, often referred to as "operations," occurs during the daylight hours when most tactical combat action occurs.

The plan produced by the night shift crew is basically a schedule which maps predicted supplies of available sorties into anticipated demands for air support for each of the corps' front-line divisions. This plan or schedule covers the daylight hours; i.e., the time from first light to last light. Information for the supply side of the plan comes primarily from the Air Force Air Tasking Order (ATO) and Operations Order (OO). Information for the demand side of the plan comes primarily from the Army Corps' plans and from intelligence sources. The Army has the prerogative of determining the "priority of fire," which is defined in percentage allocations of the total available sorties to each of the front-line divisions. The ASOC determines the details of timeframes, squadron assignments, and tasking modes.

Plan execution is essentially the responsibility of the day shift crew. The day crew uses the plan as a baseline which usually has to be modified in view of the most recent information regarding the available assets and the battlefield situation. The plan is adjusted and readjusted throughout the day in response to changes in the tactical situation and as preparation for anticipated developments in the near future. The

[1]This reprieve that may soon disappear with the introduction of technologies that enable ground and air-to-ground combat during night and low visibility conditions.

day crew therefore deals both with the present and with the near future, and with actual and anticipated supply and demand problems. The key to successful air support service from the ASOC to the corps is to manage sortie generation and sortie expenditure such that the corps receives optimal support in the present, throughout the day, and over the foreseeable length of a conflict.

The differing roles of the night and day crews at the ASOC reflect a partition into Planning and Operations (plan execution) that is common throughout the Air Force Tactical Command and Control System. The ASOC therefore can be considered a *representative* subdomain of the offensive side of tactical command and control.

Within the ASOC domain, this project focused on the decision problems faced by the dayshift; i.e., by operations personnel. The decision problems involved in planning have already been investigated in detail through projects like TAOTTS (Barnthouse, 1989) and TEMPLAR (McCune, 1985; Priest, 1986), which provide training and decision support to the process of developing Air Tasking Orders.

It is important to note that the decision problems faced by planners and operations personnel are quite dissimilar. One important distinction is the time available for solving decision problems. Planners usually have much more time than do operations personnel. Planners generally concern themselves only with a relatively distant future; operations personnel must alternate between the present and the near future. Planners decide on a global mapping of sorties into demands, not the details of mission implementation and coordination. Operations personnel must deal with very specific assignments of sorties to very specific requests, and they have to deal with the details of mission implementation and coordination.

THE ASOC OPERATIONS TEAM: MEMBERS AND THEIR FUNCTIONS

The ASOC operations team consists of six officers and three noncommissioned officers (NCOs) called "technicians." The NCOs share in the officers' workload of decision-making tasks and implement the officers' orders and provide other assistance as needed. Officer positions and duties are as follows:

Director:

> Oversees entire ASOC operation. Primarily concerned with overall situation assessment and advance planning. Issues guidelines for operation to rest of team. May handle support of large or critical Army operations personally.

Fighter Duty Officer I (and Technician):

> Assists the Director in situation assessment and planning. Assigns air resources to air requests. Coordinates mission support requirements between Army and Air Force in conjunction with G-3 Air.

Fighter Duty Officer II (and Technician):

Implements air support missions as directed by Fighter Duty Officer I.

Reconnaissance Duty Officer:

Assigns air reconnaissance resources to reconnaissance requests.

G-3 Air:

Army officer. Functions as direct liaison between ASOC and the ARMY Corps' G-3. Prioritizes requests for air support. Determines which requests can be handled by "organic" (Army) resources.

Intelligence Officer (and Technician):

Verifies requests, provides information regarding other targets in same area as request, and provides information regarding air defense threats.

The entire team works in a fairly cramped quarters.

DECISION MAKING DURING ASOC OPERATIONS

Several specific examples of the types of decision problems which occur during ASOC operations are given below. The following examples illustrate the type of decision-making skills that need to be trained.

Example 1:

The ASOC Fighter Duty Officer I (FDO 1) receives an air request to neutralize eight tanks within the next hour. The request has been verified by the appropriate officers.

The FDO I must now decide which of the available airplane assets should be tasked to satisfy this request. If he has no assets to task, he must refuse the request. If weather or threats make mission success questionable, he has to weigh risks against benefits.

Example 2:

The ASOC receives a Corps request to neutralize a large number of tanks attacking along a highway.

This type of request generally requires tasking of a fairly large number of airplanes, some of which will be the actual attackers and some of which will fly support missions. This type of request may include participating Army assests.

This sort of decision problem is usually handled by the ASOC Director, who decides on force composition, timing. coordination, and other relevant factors.

Example 3:

The ASOC receives weather information that one of the bases where it has airplane assets will be under zero visibility, zero ceiling conditions within an hour. The weather is estimated to clear within 2 hours.

The ASOC must now decide how to compensate for the shortfall of available sorties. This may have to involve a complete replanning effort.

In any of the examples above, the decision maker has to consider multiple factors. Some of the most important factors are pictured in Figure 3.

The common thread that runs through all three examples is the fact that the decision maker has to generate options for satisfying a current or anticipated demand and then make a choice from among

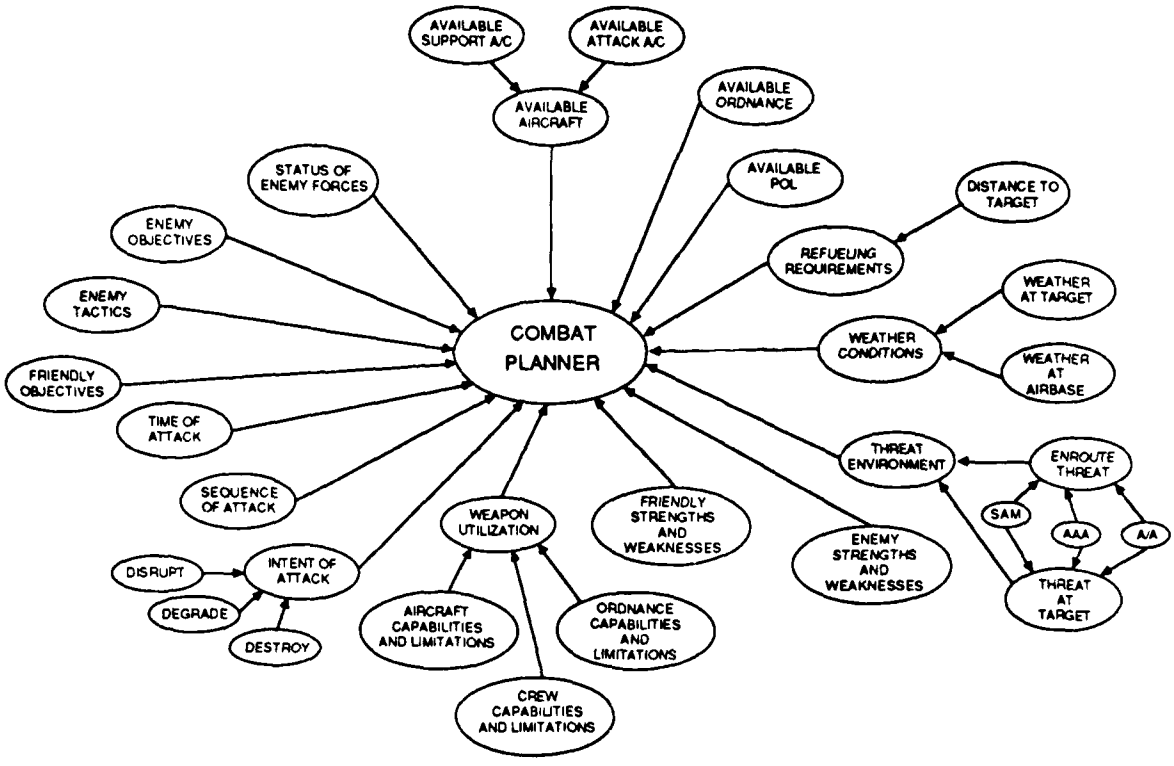

Figure 3. Factors Considered by Tactical Command and Control Decision Makers.

them. This "generate and choose" concept of decision making falls within generally accepted definitions of decision making (see Nickerson & Feehrer, 1975; also Wickens, 1984). The issue of a generic definition of decision making and the more narrow issue of defining tactical decision making is not further treated in this report. We confine ourselves here to a definition by example. The taxonomic issues surrounding decision making are more immediately related to the other part of this project, the search for a generic instructional strategy, and will be discussed in a report dedicated to this topic.

Unaided Knowledge Acquisition

The specific knowledge and heuristics that enable ASOC personnel to make the kinds of decisions illustrated above represent the domain knowledge that has to be made explicit in order to provide the substance of training these skills in others. The methods, results, and problems noted during an initial knowledge acquisition effort are described below.

METHODS

The plan in exploring the ASOC domain was to go "breadth first" and then to narrow the focus and explore in depth. This strategy was predicated on the assumption that even the relatively narrow subdomain of ASOC operations was still too broad to permit adequate coverage within the constraints of the project. To further narrow the scope, it was necessary to first achieve a broad (but fairly shallow) understanding of the ASOC domain.

The techniques used during initial knowledge elicitation consisted of focused and structured interviews (Schraagen, 1986). Both types of interviews follow some type of agenda. The difference is that focused interviews use a "breadth first" strategy, whereas structured interviews use a "depth first" strategy.

One or more knowledge engineers and one or more experts participated in each of the interviews. The interviews were tape recorded; each of the resulting scripts was analyzed by three knowledge engineers in succession. Analysis consisted of identifying knowledge elements within the often rather colloquial discourse. The knowledge elements were transferred to 3- by 5-inch cards which were then sorted and grouped under topics that emerged during the sorting process. The intent was to identify and use naturally occurring knowledge categories and not to impose any preconceived classification.

RESULTS

A quantitative description of the initial knowledge engineering effort is presented in Table 1. A total of four knowledge engineering sessions were held in roughly 4 months. Knowledge engineers and experts spent 56 hours together. These sessions produced 26 tapes and 693 pages of transcripts. Analysis of the transcripts produced 1,599 knowledge element cards. The knowledge elements have an estimated redundancy of 40% to 50%. The analysis process is labor-intensive. Productivity is estimated at about 1.5 knowledge elements per analyst hour.

The entire initial knowledge engineering effort took more than 500 hours of analyst time and produced a broad, uneven, and relatively shallow surface layer of knowledge about the ASOC domain. The analysts involved perceived the effort as inefficient and felt that there were two reasons for this inefficiency: the team's inexperience, and shortcomings inherent in the interview methods. The team had to learn how to elicit knowledge. This detracted at least initially from the productivity of the knowledge engineering sessions. Both interview methods are relatively inefficient, but focused interviews produce much more "noise" (rambling, anecdotal discourse) than do structured interviews. The structured interviews were case-based, meaning that detailed and specific tactical situations were presented to the experts. It was much easier to stay on the target topic with this type of interview, although productivity in terms of knowledge elements was not higher (see Table 1).

Table 1. Unaided Knowledge Acquisition Activities

No.	SMEs	KEs	Contact Hours	Focus and Methods	No. of Tapes	No. of Pages	No. of Cards	No. of Analyst Hours
1	2	1	16	Introductory background in ASOC Focused interview	6	156	604	120
2	1	1	16	ASOC communications, job positions, information flow Focused interview	7	182	361	150
3	1	3	8	Constructing "special packages" Structured interview	7	233	384	160
4	3	2	16	Handling of ATRs, effect of weather, troop movements Structured interview	6	122	250	120
Totals	7	7	56		26	693	1,599	550

KEs = KNOWLEDGE ENGINEERS

Case-based interview methods did have one significant drawback: The "case," which consisted of a script of events in a defined tactical environment, would fall apart if the expert made a decision at some point that would make the subsequent events in the script improbable or impossible. This type of failure indicates the need for some minimal capability to adjust the interview script to unexpected expert reactions. The fallibility of the knowledge engineering script is a problem that plagues all pre-scripted exercises. As mentioned previously in Section 1, recovery from such deviations in exercises is a very labor-intensive process which has to take place during a time-out or at night.

The inefficiency of the knowledge engineering process is well known as the "bottleneck problem" of expert systems technology (Hayes-Roth & Waterman, 1984). In addition to the inherent inefficiency of current methodologies, a number of other problems were identified which inevitably accompany the process and which, if left unattended, can jeopardize the success of a knowledge engineering effort. These problems are described in some detail below.

KNOWLEDGE ENGINEERING PROBLEMS

Five types of problems were identified by the team. None of these are new; they have all been noted and described before by other researchers who have engaged in knowledge acquisition efforts. The problems have been referred to by a variety of names; in the present report, they are referred to as follows:

1. UNCERTAINTY problem
2. EXPERT PARADOX
3. CATCH 22 problem

4. ACCESS problem
5. QUID-PRO-QUO problem.

The UNCERTAINTY problem refers to the fact that domain knowledge is often rule-of-thumb, nondeterministic, and of unknown optimality. Tactical decision problems often have multiple feasible and acceptable solutions. The solutions may differ in "quality," but it is quite difficult to find generic metrics to assess solution quality. Experts may display a low degree of consensus both as to which of several feasible solutions is best and as to the criteria that should be used to determine the best solution.

It is extremly difficult to identify optimal solutions in the tactical environment. There are many reasons for this. In addition to individual differences, there are many factors which are unknown, unpredictable, and fuzzy. A particular solution may optimize effectiveness in a local instance but cripple *overall* effectiveness. Knowledge engineering in the tactical environment therefore should seek to identify a set of plausible and reasonable solutions and define their relation to both local and global factors. The knowledge engineer is never certain that the solution chosen is best or would "win the war." The solution is not testable in any real sense.

The EXPERT PARADOX problem consists of the fact that performers who are "merely" competent can often better verbalize the reasons for their decisions than those performers who are acknowledged as true experts. Competent performers also show generally less irritation than do experts when asked questions by a knowledge engineer. According to Waterman (1986): "The more competent domain experts become, the less able they are to describe the knowledge they use to solve problems!" Waterman (1986) referred to this phenomenon as "the knowledge engineering paradox" and said that the domain expert "... makes complex judgments rapidly, without laboriously reexamining and restating each step in his reasoning process. The pieces of basic knowledge are assumed and are combined so quickly that it is difficult for him to describe the process." Similarly, Johnson (1983) reported that "... paradoxically, an increase in expertise seems to result in a decline in ability to express knowledge." Johnson called it "the paradox of expertise." Along these lines, Fraser (1987) pointed out that "knowledge acquirers may provoke resentment by rejecting the expert's description of his reasoning and pressing him to justify every conclusion."

We had access to several experts who differed as to both their levels of expertise and their verbal skills. It appears that experts with higher levels of expertise operate in a nonverbal, intuitive mode. When they are queried for the reasoning that led them to a particular decision, they are basically forced to invent a plausible story. Doing so requires them to revert to an effortful analytical mode that they have long ago left behind. They appear to feel insecure in this mode. Insecurity is aggravated when the knowledge

engineer attempts to probe inconsistencies in successive stories. As a result, the expert frequently gets irritated or avoids direct answers by digressing into more or less pertinent anecdotes.

In contrast, performers at lower levels of expertise appear to operate in an analytical and more effortful mode. They seem to have traces of their reasoning readily available and, given good verbal abilities, produce without hesitation fairly detailed accounts of their decision processes. They exhibit less of a tendency to become irritated or to digress.

The knowledge engineer must be especially concerned with the CATCH 22 problem. A knowledge engineer is caught in "Catch 22 situation" in that he cannot elicit deep knowledge unless he asks the right questions, but he cannot ask the right questions unless he has some deep knowledge. Fraser (1987), referring to this as a "chicken and egg dilemma," pointed out that, "It is difficult for the knowledge acquirer to achieve the competence required to elicit, and meaningfully interpret, the knowledge that experts convey."

In our case, the problem manifested itself in the form of increasing redundancy in expert responses. After the first two focused interviews, the subsequent structured interviews produced less new and deeper knowledge than was expected. Because the structured interviews were based on specific hypothetical decision "cases," it was expected that the experts would produce not only specific decision responses but also specific, detailed, and deep reasoning explaining their decisions. These hoped-for responses were not forthcoming. Instead, much of what was learned during the last two interviews was redundant with knowledge that had already been acquired during the first two interviews.

The ACCESS problem refers to the fact that domain experts, being experts, are usually quite busy and often remotely located. Schraagen (1986) listed "the expert is inaccessible" as a common complaint of knowledge engineers. Waterman (1986) said: "Pick a nearby expert, preferably in the same city. Otherwise, consider relocating the expert for the duration of the project." Such a solution, however, is often not practical even if feasible.

That was and is true in our case. Our situation was and is aggravated by the geographical distance between the knowledge engineers who live in Southern California and the experts who live in the heart of Texas. Commitments on either side, as well as travel restrictions, have limited access to experts in the past and are expected to impact the project in the future.

The QUID-PRO-QUO problem is related to the ACCESS problem. Knowledge acquisition activities take experts away from their primary jobs. Any unit who makes experts available therefore "donates" often substantial amounts of their best people's time and, because knowledge engineering is hard work,

these SMEs are not necessarily having a good time at it. As a result, KEs may face decreasing motivation on the part of the SMEs or the SMEs' unit to support a project whose eventual benefits are difficult to visualize. In addition, Schraagen (1986) stated that "experts may be afraid of losing their jobs, of being replaced by computers, or they may be skeptical about the value of using artificial intelligence and computers." The solution to the problem is a quid-pro-quo which is truly useful to the experts and/or their unit. "Give the expert something useful on the way to building a large system," said Hayes-Roth and Waterman (1984). To be perceived as "useful," a benefit must be immediate or near term, so that personnel who are currently at the unit will be able to make use of it before they get transferred. Without some type of near-term and practical quid-pro-quo arrangement, it is unlikely that a productive relationship can be maintained throughout the project.

Computer-Aided Knowledge Acquisition

CONCEPT

The results of the unaided knowledge acquisition effort pointed to a need to improve the effectiveness and efficiency of further knowledge acquisition efforts for the project.

The UNCERTAINTY problem arises from inherent characteristics of the domain and can basically be addressed only by abandoning the idea that every decision problem has to have a solution that is in some sense optimal. The EXPERT PARADOX problem is easily overcome if SMEs are experienced enough to be competent, but not so experienced that they have already made the transition to fast, parallel, and intuitive processing. That leaves the CATCH 22, the ACCESS, and the QUID-PRO-QUO problems as targets for a different type of knowledge acquisition methodology. To solve these problems, the following functional requirements had to be satisfied:

1. A convenient case generator mechanism must be available which enables SMEs to employ their domain knowledge to construct the kind of rich and realistic decision scenarios which the knowledge engineers could not construct due to their lack of sufficiently deep knowledge.
2. The case generator must be capable of presenting SME-constructed decision scenarios to experts (other SMEs), permit them to perform the same information search processes they use in reality prior to making a decision, accept the expert's decision input, and provide a means to record the expert's reasoning.
3. The case generator must reside on hardware available to the experts and should not require the presence of an analyst/knowledge engineer for scenario construction or for scenario presentation.

The first requirement addresses the CATCH 22 problem by employing the expert's deep knowledge to create the knowledge elicitation stimuli. The second requirement generates a system that can function as a knowledge acquisition device and/or as a training device. By providing for a training device, the requirement addresses the QUID-PRO-QUO problem. Together with the third requirement, it also addresses the ACCESS problem, by making a convenient system available at the expert's site that would enable a knowledge elicitation process without requiring the presence of a knowledge engineer.

The purpose of knowledge elicitation was to acquire the instructional content for the training system prototypes. The case generator system therefore had to have the capability to capture the decision-making process in explicit form. To get a complete picture of the expert's cognitive processes during decision making, it was necessary to capture the expert's decision output and his underlying reasoning, *as well as* a trace of his overt, observable activities in arriving at a decision. The system environment therefore had to permit the expert to engage in a pattern of interaction with the system that would, in all functional respects, be identical to the real pattern of interaction between the expert and the operational environment. For example, if the expert during actual operational decision making would have to query specific sources of information, he should be able to make the same kinds of queries in the desired case generator system and receive the same kinds of information from the system as he would from actual information sources. The system environment therefore had to be designed such that it would be at least functionally isomorphic to the operational environment.

Functional fidelity would, of course, also benefit the intended use of the system as a training device. The question was whether training applications of the system would require not only functional fidelity but also some degree of physical fidelity, such as precise replication of paper forms, status board layouts, and message formats. The answer to this question is basically dependent on the training objectives to be supported by the system. If the system is to support the acquisition of procedural skills, then physical fidelity is indeed required. If the system is to support the acquisition of decision-making skills, then functional fidelity is likely to be sufficient. If the system is to support both types of objectives, then both functional and physical fidelity should be built into the system. As it appeared desirable to build a system which would support a variety of training objectives, it was decided to replicate physical aspects of the operational environment to the extent it was possible to do so without detracting from the primary task of developing a functional case generator system for knowledge elicitation purposes.

IMPLEMENTATION: KEATS

The functional requirements delineated above formed the basis for a system design which was implemented in Smalltalk V 286 on a Zenith 248 microcomputer. Because this system supports both knowledge engineering and training activities, it is therefore called the Knowledge Engineering And Training System (KEATS). The operating modes and design features of the system are described below.

Operating Modes

KEATS has five operating modes which are accessed through the system's main menu (see Figure 4 below).Two of the modes are designed to facilitate knowledge acquisition in the study domain: the BUILDING mode and the KNOWLEDGE ENGINEERING mode. Besides these two modes, KEATS has a TRAINING, a REVIEWING, and an ADMINISTRATING mode. The TRAINING mode is identical to the KNOWLEDGE ENGINEERING mode but does not include the automatic knowledge elicitation queries. The REVIEWING mode allows on-line review and critique of transcripts collected during KNOWLEDGE ENGINEERING sessions. The ADMINISTRATING mode is used to keep track of exercises, transcripts, and system users. The BUILDING, KNOWLEDGE ENGINEERING, and TRAINING modes are discussed in detail below.

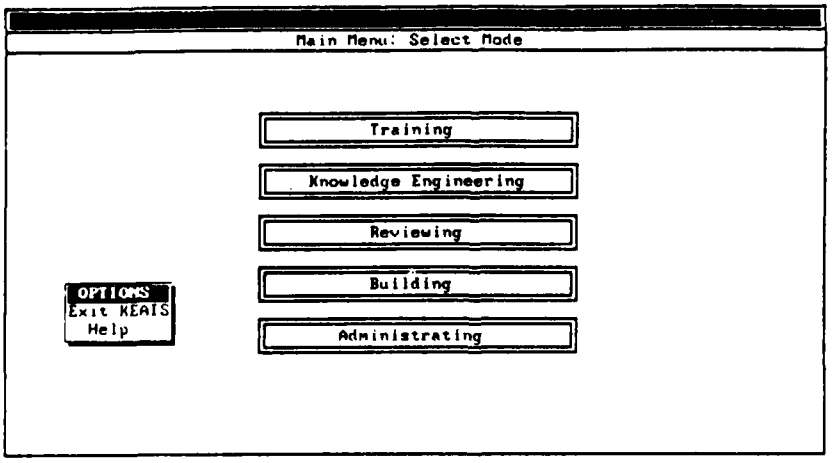

Figure 4. KEATS Main Menu.

BUILDING Mode. This mode is available in GUIDED and UNGUIDED submodes. The GUIDED submode is intended for a novice user. It is less elegant and less efficient to use than the UNGUIDED mode, but it is easier to learn and protects the user from errors. In either BUILDING mode, the user constructs an ASOC *exercise* which typically covers the daylight hours of 1 day. Each exercise consists of two parts: a *tactical situation* and a *script* of events (see Figure 5).

Events present decision problems. The tactical situation provides the context within which the decision problems occur. Tactical situations and scripts are developed with two separate editors. Both editors work along the same principle: The user selects objects from a collection of standard objects and customizes them for the exercise. For example, as a part of defining the tactical situation, the user may select a standard armor division which he customizes by giving it a name, a location, and a current combat strength; as a part of defining the script, the user may choose an air request event which is instantiated by a specific origin, a target, and a desired time-over-target.

The definition of the tactical situation must contain a minimum set of mandatory scenario objects in order for an exercise to run in the TRAINING or KNOWLEDGE ENGINEERING modes. A special exercise validation routine is available which ensures that the minimum set is present.

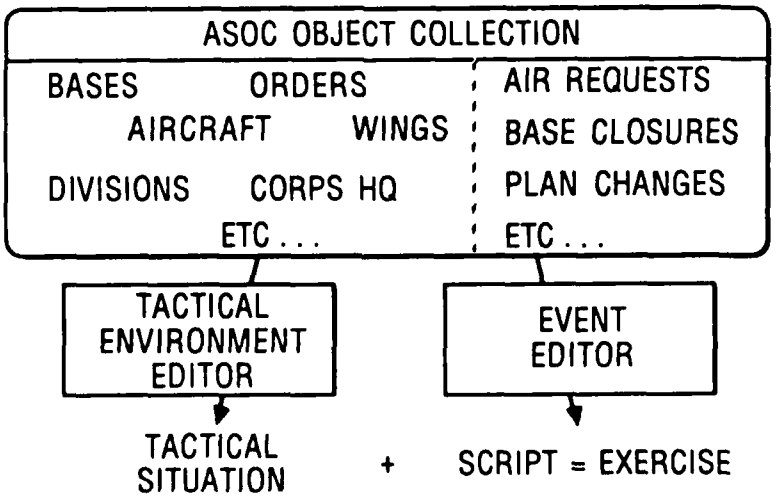

Figure 5. KEATS Building Mode.

The result of exercise building is thus a chain of events which occurs in a specified tactical environment. Any tactical situation can be combined with any number of different scripts. Any defined tactical situation and script can be edited. Once a small number of different exercises exists, additional exercises can be built very efficiently by modifying the existing ones.

KNOWLEDGE ENGINEERING Mode. In this mode, the user "runs" a previously built exercise. Exercises unfold in two phases: an initial orientation phase and a subsequent operations phase. The general flow of an exercise is shown in Figure 6.

During the *Orientation Phase* of an exercise, the user first familiarizes himself with the overall tactical situation. This initial step simulates the beginning of the day shift in the real world. This familiarization process is enabled by making available the same information resources that are present in the real world; i.e., by supplying briefings and orders to review, by enabling question-and-answer interactions with members of the ASOC team or with external agencies, and by reviewing status boards and maps (maps are currently supplied on paper).

The result of the *Orientation Phase* is an initial *Situation Assessment* which, in turn, leads to an initial *Plan Of Action*. The *Plan Of Action* usually entails preparing some of the air resources for immediate tasking by putting them on various levels of alert.

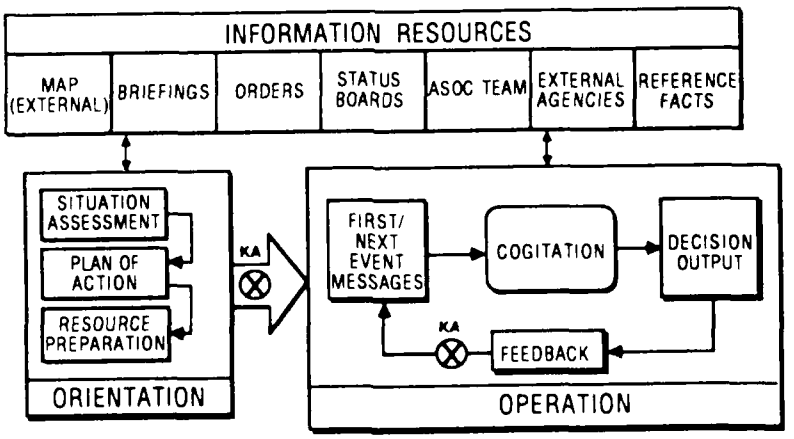

Figure 6. Exercise Flow in KEATS.

The user can take as much time as he wants for this phase. Before he can begin the second phase of the exercise, the KNOWLEDGE ENGINEERING mechanism elicits his initial situation assessment, his plan of action, and the specific conditions that led him to put certain airplanes on certain alert states. The user responds by typing the requested information into forms with a word-processor-like facility.

During the *Operations Phase* of the exercise, the user reacts to events as he triggers them. Events show up as messages which arrive at a *Message Desk*. He reacts to events as he would in the real world by accessing various information resources to confirm the event, to gather additional information about it, and to determine solution options. After this period of "cogitation," he enters a decision response to the event by telling some other agency to do something; i.e., by issuing a specific order.

KEATS evaluates the user's response by checking for violation of *physical constraints*. For example, the user may respond to a request for air support by tasking some wing to fly a mission with four airplanes of a specific type that must be at their target during a certain timeframe. KEATS checks whether these airplanes are indeed available at that wing and whether they can make the desired time-over-target (TOT). The results of constraint checking are returned to the user as feedback messages. If the tasked mission does not violate any physical constraints, the user receives a message which indicates that the wing accepts the tasking. Otherwise, the wing sends a refusal message to the user.

After the user has reacted to an event, he is queried by the KNOWLEDGE ENGINEERING mechanism -- first, with respect to the specific conditions that led him to the decision he made, and second, with respect to any changes in the situation assessment or plan of action that might have been triggered by the event. Again, he types in his responses. All user-system interactions and all user inputs are saved on a transcript which is, in fact, a detailed decision-making protocol.

The knowledge engineering queries in each exercise phase are designed to elicit knowledge from two hierarchically related levels of decision making shown in Table 2. On the *macro* level, the ASOC battle manager deals with the "big picture" by assessing (and constantly reassessing) *the overall tactical situation* for the ASOC and by deciding on (and changing) a plan of action which is consistent with this assessment. On the *micro* level, the battle manager decides on what to do about a specific event. Decisions on this level are guided *and* constrained by decisions made on the planning or macro level.

This knowledge acquisition strategy is essentially based on the two modes of decision making (i.e., planning and operations) that are found throughout the Tactical Command and Control System. The battle manager must form and continually update a global perception of the tactical situation and must

Table 2. Knowledge Elicitation at Queries in KEATS

KNOWLEDGE LEVEL	QUERIES DURING	
	ORIENTATION PHASE	OPERATION PHASE
MACRO LEVEL ("BIG PICTURE")	WRITE INITIAL SITUATION ASSESSMENT WRITE INITIAL PLAN OF ACTION	WRITE CHANGES TO SIT. ASSESSMENT WRITE CHANGES TO PLAN OF ACTION } AFTER EACH DECISION OUTPUT
MICRO LEVEL (SPECIFIC EVENT)	RESOURCE PREPARATION: (PUTTING AIRCRAFT ON ALERT) STATE CONDITIONS SHOW BOUNDARIES	IF NOT ROUTINE: STATE CONDITIONS SHOW BOUNDARIES IF ROUTINE: GIVE OPTION COMPARE } AFTER EACH DECISION OUTPUT

formulate a plan for dealing with that situation. He must solve any particular decision problem by considering the local, short-term parameters of the problem within the context of the "big picture"; i.e., his overall situation assessment and plan.

It follows then that knowledge acquisition must aim to capture both of these aspects of expert knowledge. It must capture the expert's global or *macro* perception of the tactical situation and the general heuristics for dealing with the overall situation. It must also capture the expert's local or *micro* perception of the specific decision problem (presented by a given event) and the heuristics that apply to solving decision problems at that level. Finally, knowledge acquisition must capture the interaction between context and specific problem; i.e., the reasoning that links the macro and micro levels.

The knowledge acquisition procedures currently implemented in KEATS are designed to capture these knowledge elements. KEATS queries the subject first on his overall perception of the situation, by asking him for his ASSESSMENT of the situation; it elicits general heuristics by asking for his PLAN OF ACTION. As subsequent events occur, the subject is asked to provide any changes or updates he might want to make to his ASSESSMENT or PLAN OF ACTION. To elicit knowledge on the micro level, the subject is queried after each decision for the specific CONDITIONS that have led to the decision. It is expected that this query will yield responses on both levels, micro and macro, and that it will provide insight as to how these two levels interact. This interaction is further explored with the final type of query, where the subject is asked to indicate what conditions would have to change (and how much) to make his decision invalid (i.e., to trigger a different decision).

Events may not always require a decision but rather, a "routine" or standard response. Those are typically the events that fit perfectly with the current situation assessment and plan of action. They are, in other words, expected and thus merely trigger a planned response without requiring a choice between response options. If this occurs during a knowledge engineering session, the subject can so indicate. As a result, the subject does not have to repeat a routine set of conditions to justify and explain his response. Instead, the subject is asked to provide an alternative response to the routine one and compare the two. This type of query is designed to elicit additional micro-level reasoning. It also prevents the subject from answering the knowledge engineering queries in routine cases by merely referring to answers to some earlier query.

TRAINING Mode. In the training mode as in the knowledge engineering mode, the user (or trainee in this case) goes first through the *Orientation Phase* which simulates the shift takeover in the morning. Once he has familiarized himself with the tactical situation, he can enter the *Operations Phase*.

The difference between the two modes is the absence of knowledge elicitation queries in the TRAINING mode. This has two consequences. First, user-system interaction is smoother and more natural without the constant intrusions of the "knowledge inquisitor." Secondly, the *Orientation Phase* is changed. In the TRAINING mode, the user does *not* have to develop an explicit situation assessment or plan of action during the *Orientation Phase*; nor does he have to prepare his air resources before he transitions into the *Operations Phase*. He can, if he so chooses, skip the entire *Orientation Phase* and "start the war" right away. Once in the *Operations Phase*, though, he will suffer the consequences by not being able to task airplanes which can get to their targets on time -- an example of the functional fidelity built into KEATS.

In general, training *fidelity* in KEATS is a function of what has been built into the system on the one hand and a function of what has been built into a given exercise on the other hand. The KEATS environment by itself provides a high degree of functional and physical fidelity. Status boards, tasking forms, briefings, orders, and even the interior of the ASOC command post are faithfully replicated. The trainee can communicate via ASK and TELL functions with all agencies/individuals he normally communicates with in reality, including his teammates in the command post. The communications themselves are restricted to the possibilities offered by a series of menus. These can be arbitrarily expanded until they include all standard or common inquisitory and directive communications.

Whether the particular scenario presented by KEATS appears credible and/or realistic to the user depends on how well the exercise is built; i.e., the only limitations in this respect are the skill and expertise of the exercise builder. A scenario, for example, can duplicate one that is planned for a major

FTX or CPX, or it can represent actual scenarios expected during real conflicts. This offers the possibility for using KEATS not only for training but also for *mission rehearsal* purposes -- an example of the versatility offered by the TRAINING mode.

The *versatility* of the TRAINING mode stems from the fact that KEATS is essentially a shell, rather than a fixed program. The BUILDING mode, backed by the ASOC object collection, allows the construction of training exercises which are arbitrarily complex and which can be slanted toward particular problem types that occur in the ASOC world, thus permitting training in support of a wide variety of position-specific and problem-specific training objectives. In addition, training exercises can be constructed for specific theaters. Exercises can also portray specific actual operational war plans or plans for a scheduled major CPX or FTX, thus enabling employment of KEATS as a *mission rehearsal* system.

The TRAINING mode can be employed by single users, by a trainee with an instructor or coach, or by small groups. At the present time, KEATS can only be presented at one terminal at a time; i.e., it cannot accommodate multiple interacting terminals in a network.

The *feedback* capabilities of the TRAINING mode are limited. As mentioned in the previous section, KEATS does check for violations of physical constraints. The results are fed back to the trainee in the form of messages from agencies which were affected by the trainee's decision. Checks for physical constraint violation evaluate the technical feasibility of a decision but they do not evaluate its "tactical wisdom." The latter requires the kind of deep domain knowledge that KEATS is designed to elicit. KEATS, in its present form, also provides only minimal feedback in terms of changes of the tactical situation as a result of trainee decisions. The script is largely fixed. However, some decisions do lead to modifications. For example, choosing to supply air support to a division in a particular manner causes requests on the script that would normally be sent to the ASOC to be intercepted by the division.

User Interface

An appropriate user interface was considered absolutely crucial for system success. The interface had to be tailored to the ASOC users and to the conditions under which they would use the system. The prospective ASOC user was assumed to be an officer (Captain through Lieutenant Colonel) who is comfortable with using computers but who is not a programmer. His motivation to do the extra work in supporting an outside agency was assumed to be low to nonexistent (worst case!). He would have little time to spare and therefore could not be expected to spend more than 1 or 2 hours on "trying to figure

out the system" (i.e., on learning how to operate KEATS). He would have to use the system on his own, without assistance from an analyst (knowledge engineer), a computer expert, or another user already familiar with the system.

This "usage scenario" gave rise to a number of broad design goals for the interface. The primary goal was to make the user interface as *natural* and *intuitively obvious* as possible. This is not necessarily the same as building functional and physical fidelity into the system. Functional fidelity enables the user to do all the things he must do in reality. Physical fidelity means that stimuli and tools occurring in the real world are faithfully replicated in the system. A natural and intuitively obvious interface (as we understood that notion) may include these aspects, but its primary characteristic is that it never puts the user into a quandary with respect to where he is in the system, where he can go from where he is, and what he can or cannot do when he gets there. Our goal was, in other words, to create an interface that would present little or no "navigational" or "action selection" difficulties to the user.

Another design goal was to have an interface that was as *direct* as possible; i.e., an interface that would not require system-peculiar intermediate steps before the user could perform a desired action such as look at a status board, task a mission, or ask a question.

Thirdly, it was highly desirable to make the interface as nearly "*foolproof*" as possible. This idea includes making the system almost impossible to crash, building in safeguards which prevent the user from making mistakes, and allowing for easy recovery if the user does get stuck in spite of safeguards.

Finally, the interface should, as much as possible, *eliminate trivial tasks* (such as filling in repetitive data on a status board), and it should include a means for the user to *feed back suggestions* for system improvement to the system builders.

These design goals were achieved with specific features in each of the major interface components (i.e., in System Displays, Menus, User Inputs, and Help facilities). Each of these components is discussed below.

System Displays. All screen displays in KEATS are designed to present information in a *clear and simple format*. Information which is not relevant to the current user activity is not visible on the display but can be readily accessed when it does become relevant (see discussion of menus below).

A *zoom feature* (toggled by a function key) allows the user to get a full-screen view of object descriptions and of feedback displays (see Figure 7).

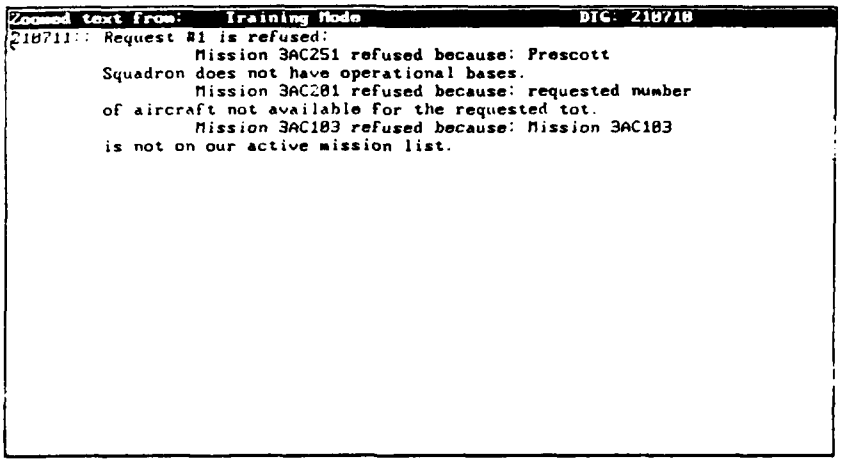

Figure 7. Zoomed Full-Screen Display in KEATS.

All screens show *orientation information* on the top line; the current operating mode is indicated by a label on the left side and by the background color of the line (green for TRAINING, orange for BUILDING, etc.) which also shows up on the main menu when the operating mode is selected. The top line also shows to the right the name of the current exercise (being built or run). Exercise time is displayed as a Date-Time-Group (DTG) at the right-hand side of the top line (during TRAINING and KNOWLEDGE ENGINEERING). Further orientation information may be displayed in the second line of the display. The third line is occasionally used to convey instructions to the user.

The knowledge engineering and training modes are designed around two *central screens*: the message desk and the ASOC screen, which are easily recognizable by their green background as opposed to the white background of all other screens. The message desk provides a running log of incoming messages and of user decisions made in response to these messages. The ASOC screen serves as a communication hub. From this screen, the user can communicate with other people inside the ASOC command post and with all necessary agencies outside.

Menus. Three types of menus are used for navigation. Full-screen windows are used for the main menu (see Figure 4) and the top menu of the BUILDING mode. The other two types of menus are pop-up windows overlaid on screen displays. Either they appear automatically or they are called up by the

user (by clicking right button on mouse). The automatic pop-up menus force the user to make selections of submodes and types of exercises. The pop-up menus called by the user are called *Options*. They display all facilities which the user can reach from his current location.

All menus either appear when called or appear automatically when intermediate selections must be made to get to a destination. At all other times, the menus are out of sight, which avoids cluttering the screen displays. In all menus, selection is made by driving the cursor over the desired selection and clicking the left mouse button.

User Inputs. User inputs are made either by convenient mouse-click selection from a pop-up menu list or by way of the keyboard. The use of list selection input has been maximized to reduce tedium and labor for the user as much as possible. Keyboard inputs of text and numbers are, however, unavoidable in the BUILDING and KNOWLEDGE ENGINEERING modes.[2]

A particularly interesting application of the list selection input mode is used for the *simulation of communications* with ASOC team members or outside agencies (see Figure 8). The user makes a series of selections which first get him the "other party," then the mode of communication which is either ASK or TELL, and then a series of topics and subtopics which allow him to construct a question or an order. The question or the order is displayed in the lower part of the screen as it is formulated. The system's reply to either the question or order appears then directly underneath.

Help. Help is always accessible from the option menus. The help text explains the selections available on the menu. Indirect help is provided by means of blocking some user inputs to prevent user error. When this occurs, a pop-up message appears which informs the user as to what to do next. A pop-up hint appears during exercise loading informing the user, "This will take a while!" This message helps to prevent user frustration and undesirable keyboard inputs during the loading process.

Overall, the user interface is completely consistent, pleasant in appearance, and, as far as limited experience shows, very easy to learn. The users who have been observed thus far were fluent in system operation after about 1 hour. Once past this initial orientation period, the users had few, if any, questions regarding system operation. The system itself had receded into the background and the decision problems posed by it became the focus of user attention.

[2]Audio recording would be available alternative for KNOWLEDGE ENGINEERING, especially if the soundtrack could be keyed to the transcript. The alternative was rejected for this first implementation because of time and cost constraints.

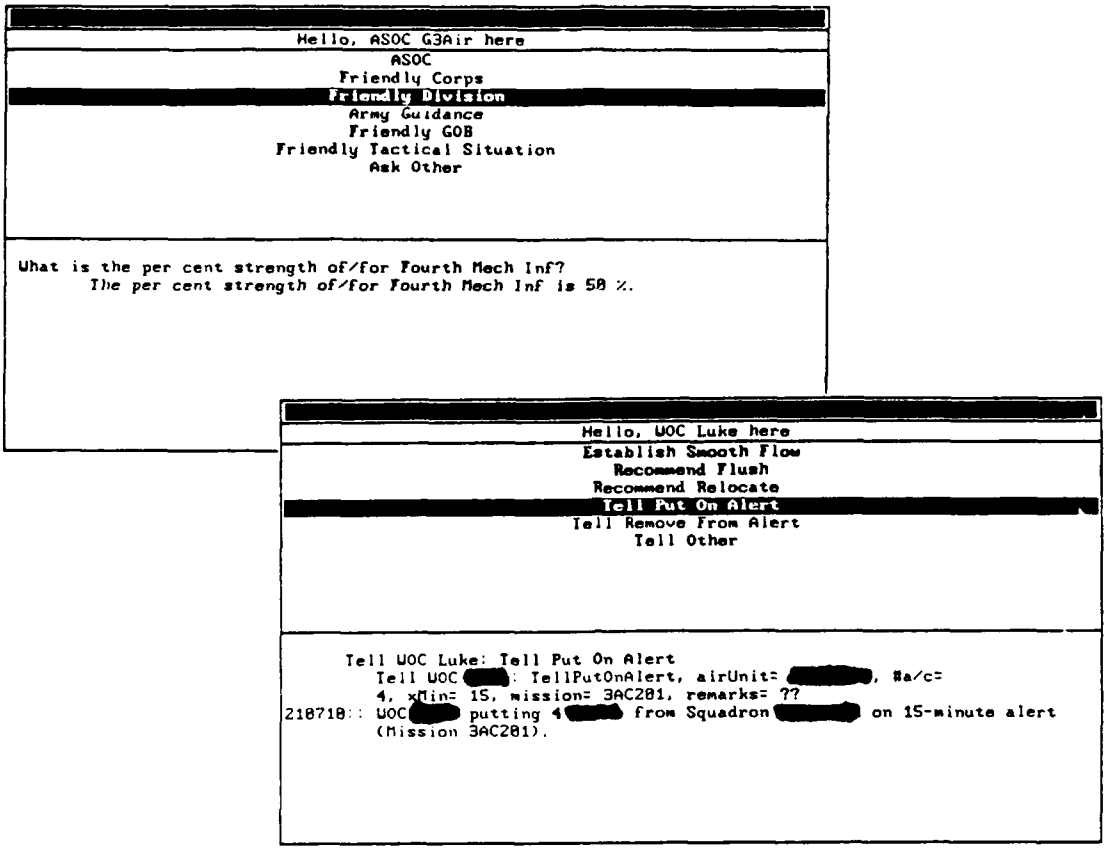

Figure 8. Ask and Tell Communications in KEATS.

Programming Environment

The functionality of KEATS as specified in the design raised several implementation issues. These included:

1. What programming approach or paradigm should be used?
2. What hardware platform should be used for development and deployment?
3. What programming language and software tools should be used?

The most important decision was to use an object-oriented approach in implementing KEATS. KEATS was to be an exploratory, fluid system; thus, a rapid prototyping software environment was crucial. Using an object-oriented approach seemed particularly suitable for this requirement. The object-oriented paradigm is based on the concept of classes and instances of classes where the behavior of an instance is specified in the class and each instance maintains its own set of local attributes. Classes are organized in a tree structure, with each class "inheriting" the behavior and variables of the class above. By adding new subclasses as leaves in the tree, or modifying the behavior of classes higher in the tree, one is able to successively refine and fine-tune an application in an especially powerful and "natural" way. If, as is usually the case, an object-oriented language comes with a large set of prebuilt classes, then the development effort is noticeably reduced.

The choice of hardware platform was driven primarily by cost, portability, and the ability to upgrade the system as more resources were required. Cost constraints required that the hardware be an IBM AT or Macintosh II class machine. Portability favored the AT (or compatible) and Mac II because of their large user base; however, the AT was favored because of its availability at the ASOC and most Air Force squadrons. Both could be upgraded in memory and mass storage; however, it was believed that the AT would provide a clearer migration path to increasingly powerful 80386 machines for greater processing speed. Thus the AT was selected; this, in turn, influenced the choice of software tools for the implementation.

A key issue in language selection was how naturally the problem domain would be represented in the language. As is true for Tactical Command and Control Centers in general, the ASOC world is composed of many elements interacting together and communicating via messages. There would clearly be advantages to being able to correlate these with corresponding data types and control constructs in the implementation language.

It was also desired that the software development environment provide a toolkit for fast construction of menus, windows, and graphics images to experiment with a variety of user interfaces.

The candidate languages were initially C, LISP, and Smalltalk. Two variants of C (Objective C and C++) provide good support for object-oriented programming but few tools for interface development. Furthermore, the object-oriented support is more in the nature of an added layer of functionality rather than an intrinsic part of the language. More telling is that they are compiled languages and not particularly suited for rapid prototyping in contrast to LISP and Smalltalk. LISP (by which Common LISP is assumed) is the most flexible of the candidates, and packages such as FLAVORS can be purchased to provide the necessary functions for object-oriented programming. However, execution speed was a significant concern, user interfaces and graphics tools were relatively spartan, and a full Common Lisp was not commercially available at the start of the KEATS implementation effort. Expense was another concern, as it would have been necessary to purchase licenses for "partial" Common LISPs.

Smalltalk is the prototypical object-oriented language/environment and is a mature and stable product with support from multiple vendors. The language has an elegant syntactical and semantic consistency and provides the most natural means of expressing the object-oriented paradigm. This is not surprising in that object-oriented programming and Smalltalk were developed hand in hand at Xerox PARC.

Although a full Smalltalk-80 was not available on the AT, a relatively complete subset was obtainable from Digitalk Inc. Their product, Smalltalk/V and later Smalltalk/V286, had been evaluated on a previous project and found to be well built and well supported.

The only drawback of any consequence to choosing Smalltalk/V286 is that this particular implementation of the language departs in some areas from a "true" Smalltalk-80 and thus is not immediately portable to other systems supporting Smalltalk-80. The differences, however, could be easily fixed (mostly changing names of a few classes, adding some infrequently used methods, changing ':=' to '<-' in the syntax).

In conclusion, an object-oriented approach was chosen because of its conceptual similarity to the problem domain, which was naturally modeled as the creation and manipulation of physical objects. Smalltalk/V286, in particular, was chosen because of its rich set of predefined data types, extensive graphical interface, availability on ATs, and low cost.

System Software Architecture

The structured interviews employed during the unaided initial knowledge engineering effort relied on *paper-based cases to acquire expert knowledge*. A case began with a description of the battlefield environment. Experts received a list of available air assets and a written briefing on friendly and enemy force strength, position, and intentions. When the experts were ready, a knowledge engineer announced events from an agenda of events (Figure 9). These pre-scripted time-ordered events were reflected in messages reporting about division movements, base closures, aircraft losses, requests for air support, and other relevant incidents. Knowledge engineers recorded the experts' reactions to the event and also attempted to accurately account for asset and battlefield state changes.

KEATS is an automated and extended metaphor for the paper-based process. KEATS implements the concepts of a case, an environment, an agenda, and a knowledge engineer using an object-oriented, agenda-based architecture:

1. A Case Librarian maintains cases
2. Environment objects store information and account for state changes
3. Event objects trigger Environment state changes
4. An Agenda Manager object sequences through Events stored on an Agenda
5. A window and menu-based user interface display information; the interface also prompts and records expert input.

AGENDA

0700	Airfield A remains inoperational with 0-0 conditions.
0705	Station 1 reports two aircraft collide while being moved. Both suffer wing damage. Estimate 10 hours to repair.
0710	First Air Cav request
0715	First Air Cav request Second Armor request
0800	Intel reports that enemy force 1 has occupied Mtn A Intel reports sighting of enemy force 4 SE of Station 2
0805	Station 3 reports loss of aircraft
0900	Base X closed down by unknown agents
0905	Station 1 reports loss of aircraft
0925	Station 1 reports loss of aircraft
1000	Base X operational again
1005	Intel reports that enemy force 3 has captured Airport B
1005	Station 4 reports loss of ? aircraft
1010	Intel reports that enemy force 3 has captured the town near Station 1
1012	Fourth Medical request
1015	Intel reports that the first friendly force has advanced to the intersection of X and Y
1020	Corps changes fire priorities to: First friendly force 30% Second friendly force 10% Fourth friendly force 60%
1100	Station 1 closed down by thundershowers
1115	Airfield A reports loss of 1 aircraft

Figure 9. Sample Paper-Based Case: Fictitious Agenda from a Fictitious Scenario.

Each of these architectural components is implemented as a Smalltalk class within the KEATS environment. The classes are structured to take advantage of Smalltalk's inheritance features. Figure 10 shows the relationship among the major architectural components, how they interact via messages with each other and with major objects built into the Smalltalk/V286 environment.

Case Librarian. The Case Librarian object saves and loads cases to disk files. The user builds a case using the KEATS' BUILDING mode and runs the case using the KEATS' TRAINING or KNOWLEDGE ENGINEERING mode. A runnable case consists of user-defined Environment and Event objects.

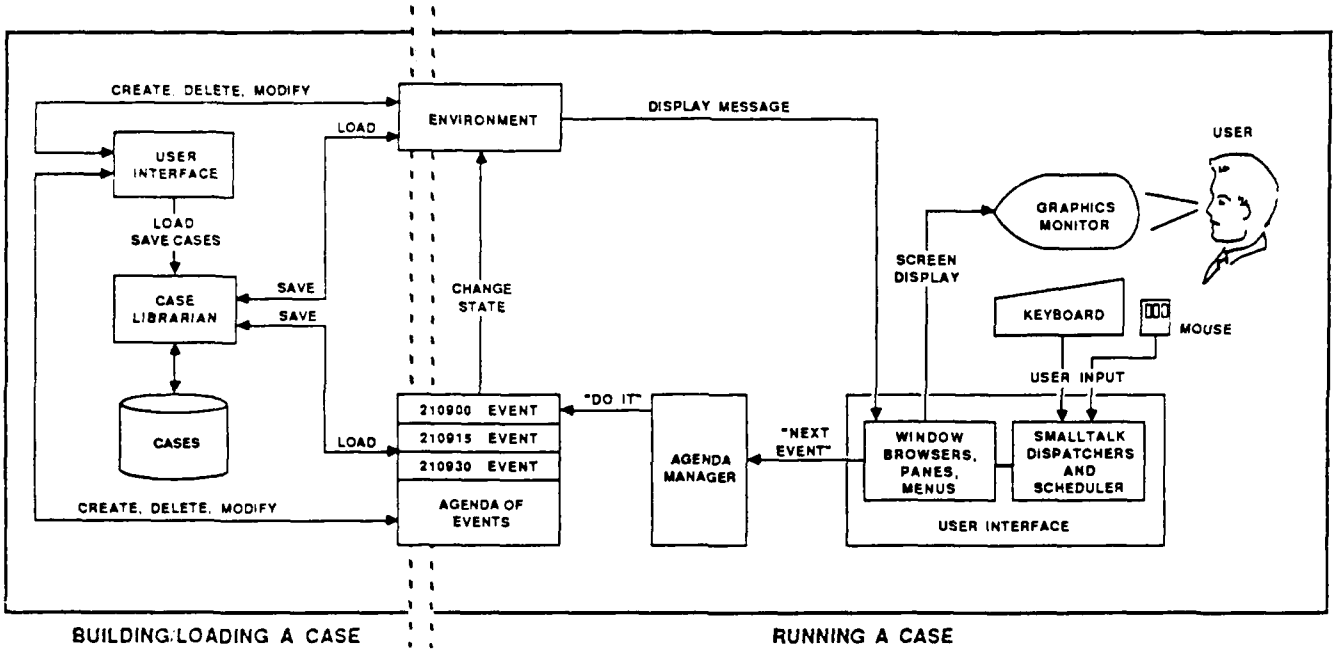

Figure 10. KEATS Architectural Components for TRAINING and KNOWLEDGE ENGINEERING.

Environment. Environment objects represent physical and conceptual objects in the ASOC's world. The ASOC tasks aircraft for Army air support. Aircraft and Army divisions are Environment objects. The ASOC is cognizant of the tactical situation. Enemy Tactical Situation, Friendly Tactical Situation, Air Order of Battle, and Ground Order of Battle are Environment objects. The entire set of classes of Environment objects is listed in the Appendix.

During case building, the user defines individual objects that belong to these classes. Each of these individual objects, called "instances," represents a unique set of values. The user defines both the quantity of instances and the values which characterize each instance. For example, an aircraft has an aircraft type, an assigned base, and a current status. The Case Librarian saves the Environment objects defined for a given case, along with the events for that case.

Events. As a case is run, the characteristics of individual objects are changed by events. Lose Fighter Aircraft is such an event. It triggers aircraft losses. During the BUILDING mode, the user may decide to reduce aircraft assets due to a ground missile attack at 9:00 a.m. He does so by creating a time-tagged event for 0900, specifying the squadron losing the aircraft, how many aircraft to lose, the reason, and the location (air or ground).

A complete list of all classes of events is found in the Appendix. The user may only author instances of Scenario Events. Other classes of events support data collection, system updates, and user changes to the Environment while running a case.

Scenario Event instances are saved as a list and sorted by time. The list, called an Agenda, is stored with the Environment instances for the case. When triggered by the Agenda Manager, Scenario Event instances send messages to Environment instances.

Agenda Manager and Agenda. The Agenda Manager runs a case for Training or Knowledge Engineering. It retrieves event instances from the Agenda and sends a message, "do it," to trigger the event. Depending on the class of event and the time tag, the Agenda Manager may sequence through several events upon activation. The KEATS user interface activates the Agenda Manager.

KEATS: KEATS Interface, KEATS User, KEATS Utilities. In keeping with the standard Smalltalk interface, the KEATS user interface is window- and menu-oriented. KEATS appears to the user as a series of full-screen windows that are sequenced from one to another either via pop-up menu selections or window option selections. From the implementation point of view, each window corresponds to a KEATS Window Browser with a predefined appearance and functionality.

The KEATS Interface Classes are found in the Appendix also. Each KEATS window that a user sees corresponds to an instance of a subclass listed under KEATS Window Browser. Application-tailored pop-up menus, data entry pop-ups, user input parameters descriptions, and user privilege descriptions complete the set of KEATS classes implemented in the present effort to develop the user interface. For example, a Message Desk Browser instance represents the Message Desk window. In the class description of the Message Desk Browser, the appearance of the window is described as a set of three window panes that compose the window: a header pane with mode/time/exercise information, a title pane, and a message display pane.

Each window pane has a corresponding user input manager called a "dispatcher." Whenever the user enters keystrokes or mouse button actions, the Smalltalk scheduler notifies the responsible dispatcher to process the input. So, when the user clicks right within the bottom-most pane, the dispatcher for that pane performs the predefined right-button action, which is to show the pop-up menu. Once the user selects an option from the menu, the selection is processed by the browser.

Selection of the ASOC screen option causes the Message Desk Browser instance to schedule display of the ASOC Screen Browser instance, relinquishing control to the Smalltalk scheduler to show the ASOC window. All sequencing from window to window in KEATS is done similarly.

Understanding the Whole. Figure 10 can be used to trace through the major processing steps for a particular event (Lose Aircraft). Beginning with the Message Desk, the user selects the Next Event option from the pop-up menu. The pop-up menu object sends a "next event" message to the Message Desk. The Message Desk responds to the "next event" message by sending a "next event" message to the Agenda Manager. The Agenda Manager retrieves the next event from the Agenda, the 9:00 a.m. Lose Aircraft event. The Agenda Manager forwards the clock to 0900 and sends a "do it" message to the Lose Aircraft event.

The event sends a message to Station 1 to lose a ground-based aircraft due to a base attack. The Station 1 squadron, an Environment object, sets the status of one of its aircraft to unavailable and sends a message to its operations center to report the loss.

The processing cycle for the event is completed when the operations center reports the loss via a message back to the Message Desk. The Message Desk updates the screen to display the aircraft loss message.

As the processing cycle illustrates, the object-oriented approach helps achieve congruity between the architectural components and the elements of case-based knowledge acquisition. This congruence and the inherent modularity of objects expedited rapid, incremental, iterative development of the KEATS metaphor. The Environment objects' close correspondence to the ASOC's world aided provision of rational responses regarding air asset status. Furthermore, the Agenda Manager, Agenda, and Events proved invaluable as simulation building blocks, triggering realistic behavior in addition to scripted events.

Technical Data

Pertinent data describing hardware requirements for KEATS, as well as the sizes of the various software components, are listed in Table 3. The system is currently installed on a Zenith 248 microcomputer at the 712th ASOC Squadron. Installation required expansion of the existing Random Access Memory (RAM) by 2 megabytes (MB).

Table 3. KEATS Hardware and Software Requirements

SOFTWARE	KEATS APPLICATION	0.8 MB IMAGE
	TYPICAL EXERCISE	0.2 MB
	SMALLTALK/V286	0.5MB
	SMALLTALK/V GOODIES	
HARDWARE	IBM PC/AT COMPATIBLE, 286 OR 386, 3 MB RAM	
	HARD DISK, 5 1/4" FLOPPY	
	EGA, MOUSE SYSTEM COMPATIBLE MOUSE	
	IBM GRAPHICS PRINTER OR HP LASTER JET PLUS	
	DOS 3.2 OR 3.3	

Summary

KEATS is basically a shell designed to facilitate knowledge engineering in the ASOC domain. It contains standard versions of all the objects in the ASOC world. Editing facilities allow a user to customize these standard objects and to construct realistic and specific decision-making scenarios. The scenarios can be presented to experts for knowledge engineering purposes, to students for training purposes, or to operational teams for mission rehearsal purposes. In the knowledge engineering mode, two levels of knowledge are elicited: knowledge used in overall situation assessment and planning (macro level) and knowledge used in reacting to particular tactical events (micro level). The expert responds to knowledge elicitation questions by keyboard inputs. The system collects a complete decision-making protocol. KEATS provides limited feedback regarding decision feasibility and has a limited capability to adjust a set of prescripted events to user decisions. KEATS in its present form is a research prototype of a new multi-purpose technology which is based on inexpensive software and hardware. It can be expanded into neighboring domains by extending the resident object pool.

III. RESULTS

At this writing, KEATS has been used only once and only in its role as a knowledge acquisition support tool. This first employment had the dual purpose of knowledge acquisition for the follow-on system and formative evaluation for KEATS. The methods used during this first trial and the results obtained are reported here.

Method

Two ASOC experts used the system for 2 consecutive days at the contractor's facility. Each expert was independently (in separate rooms) subjected to two different exercises that had been developed by the contractor, rather than by the experts themselves. On Day 1, Expert A ran Exercise 1; Expert B ran Exercise 2. On Day 2, the experts switched exercises so that both experts did both exercises on different days. Each expert was assisted during both days by a contractor analyst/knowledge engineer. The analysts were there to observe the experts and to provide "over-the-shoulder" assistance in system use if required. The analysts also taped any discussions with the expert and/or the expert's monologues (they were encouraged to "think out loud"). The tapes were made to determine whether significant information was getting lost during the relatively laborious keyboard input of reasoning.

Knowledge Product

Overall the two experts produced 29 pages of transcript. Expert A produced approximately seven more transcript pages than did Expert B. Expert A also processed five more events than Expert B processed during the same time (Table 4). Expert A had a prior exposure of several days to an earlier version of KEATS.

Table 4. Productivity During Knowledge Elicitation

Output	Expert A	Expert B
Events processed	10 + 7 = 17 [a]	11 + 11 = 22
Number of pages of transcripts	6 + 5 = 11	9 + 9 = 18

[a] Exercise 1 + Exercise 2 + Total

MACRO-LEVEL KNOWLEDGE

During the *Orientation Phase* of the exercises, the two experts produced a total of four Initial Situation Assessments (ISAs) and four Plans of Action (POAs). ISA and POA length was about six lines on the average; i.e., the length of an average paragraph of text. Length differences between ISAs and POAs or between experts were negligible.

Analysis of the assessments and plans revealed common content elements across exercises and experts. Assessments consisted of three parts: statements defining the probable *demand* for air support, statements evaluating the anticipated *supply* of air resources, and identification of *replanning triggers* or factors that should be watched because they might trigger a revision of whatever plans were made (see Table 5).

Plans consisted basically of a partial mapping of air resources by squadrons to front-line divisions. The two POAs (by each of the two experts) for one of the exercises are shown in graphic form in Figure 11. They represent clearly very different ways of dealing with the same situation, but they are constructed of

Table 5. Content Elements in Situation Assessments

Element	Expert A		Expert B	
	Exercise 1	2	1	2
Demand asssessment				
ID of need[a]	X[b]	X	X	X
When		X	X	X
Division	X	X	X	X
Priority	X[c]	X	X	X
Supply assessment				
Type of a/c	X[c]	X[d]	X	X[e]
Ordnance	X[c]	X		X
Planned allocation to meet demand	X		X	X
Time to target area			X	X
Replanning triggers (e.g., Wx at 1100, end of attacking period)				
What	X	X		X[f]
When	X	X		X[f]

[a] role; i.e., offensive/defensive attacking/holding is an important factor.
[b] did not note attack time and looked at strength and fire priority.
[c] stated as action conditions.
[d] "mix."
[e] ID of distinguishing features.
[f] on notepad.

the same elements. The ISAs which were generated prior to the plans were also different but could not logically account for all plan differences. It was inferred that other macro knowledge not tapped by the current elicitation strategy in KEATS accounts for unexplained plan differences.

During the *Operations Phase* of Exercise 1, only one change to ISA and POA was noted (by Expert A). In Exercise 2, both experts changed ISA and POA twice but at different times in the exercise (i.e., in response to different stimuli). The changes always involved a revision of the initial demand assessments and resulted in a revised plan for allocation of air resources.

Assessment and planning thinking on the macro level was also found in some of the responses to queries on the micro level. One of the experts used the Notepad facility extensively and noted some of his planning and assessment thoughts there.

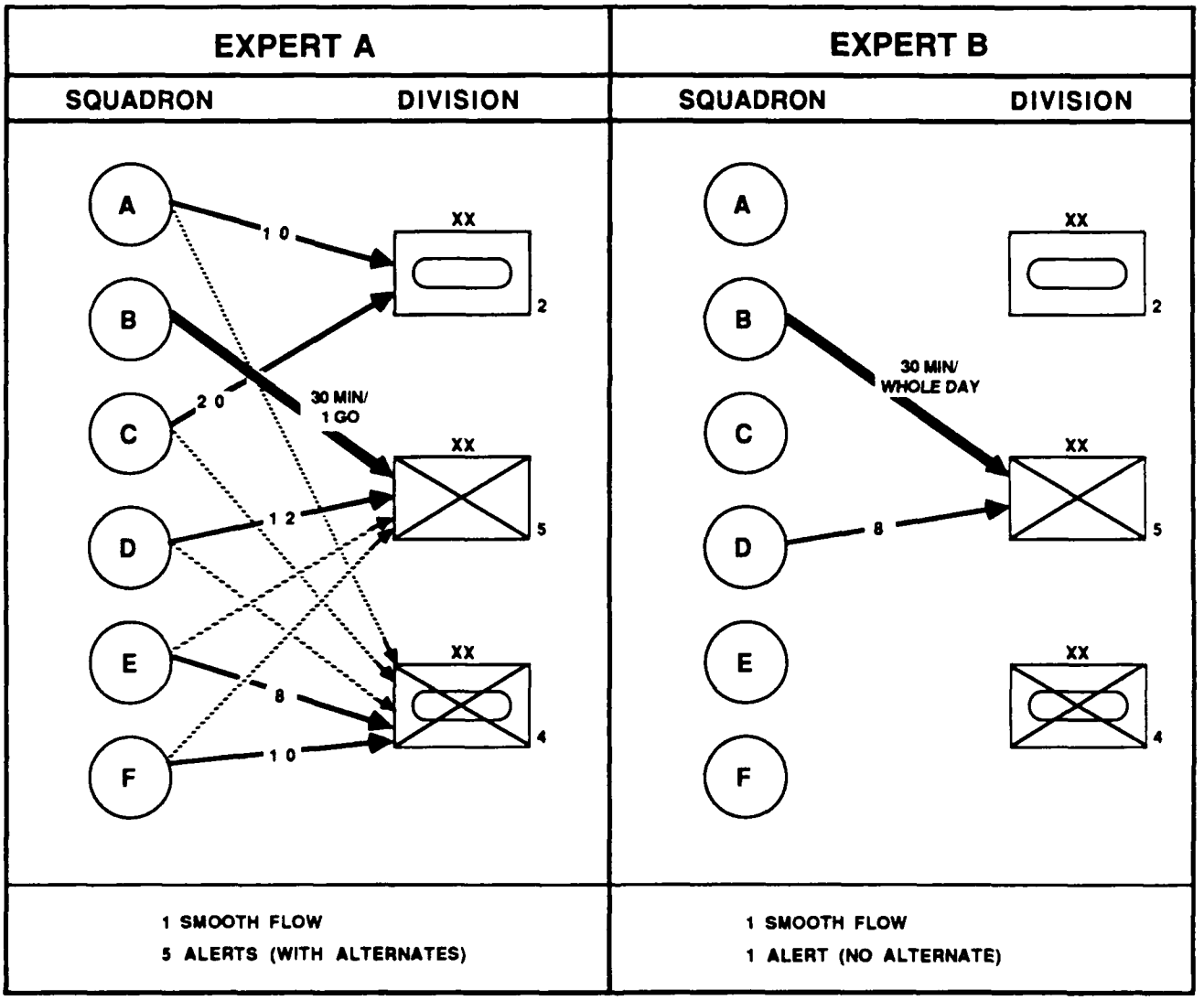

Figure 11. Plan of Action Differences in Identical Situations.

Tape transcripts and KEATS protocols contained the same relevant information. The tape transcripts also contained discourse unrelated to assessment and planning.

MICRO-LEVEL KNOWLEDGE

Micro-level knowledge was elicited for expert decisions in response to air requests. Experts were asked to list and rank-order the conditions that prompted their decisions. Altogether the experts made 28 tasking decisions and justified them by listing 135 conditions.

No differences between tape transcripts and KEATS protocols were noted, except for the presence of discourse unrelated to the decision tasks in the tape transcripts.

There were noticeable differences between the two experts. Expert A responded with an average of about four conditions, verbalized (thought out loud) his reasoning before entering the decision, and then entered his reasoning. Expert B responded with an average of about six conditions, made and entered his decision with a minimal verbalization, and then rationalized his decision post-facto.

Subsequent analysis of the tasking decisions and conditions produced 64 rules on nine different topics related to decision making in response to air requests (see Table 6).

Table 6. Number of Rules Obtained by Topic

Order of tasking	4
Requests before tasking	6
Putting aircraft on alert	2
Tasking beyond NLT	2
Selection of assets	26
Amount of support	11
Request refusals	3
Smooth flow	5
Diverts	5
TOTAL	64

Additional Knowledge Products

The first trial of KEATS as a knowledge engineering support tool was also a formative evaluation of the KEATS system. Two types of information were collected:

1. Information relating to the *fidelity of built-in functions and facilities* within the KEATS environment, and

2. Information relating to the *fidelity of exercises* that were presented to the experts (and constructed by means of KEATS' building function).

Both types of information can be viewed as knowledge engineering results in their own right. The trial produced a total of 23 items indicating changes to be made to KEATS and 15 items indicating changes to be made in the existing and in future exercises.

Cost

The cost of acquiring the knowledge product is expressed in manhours spent during the trial (i.e., during knowledge elicitation), and manhours spent after the trial (i.e., during knowledge analysis and formulation). These costs are illustrated in Table 7.

Table 7. Manhour Cost of Knowledge Acquisitions

	Experts	Analysts (Knowledge Engineers)	All
Knowledge Acquisition	29.4	29.4	
Analysis and Formulation		102.0	
Totals	29.4	131.4	160.8

Summary

These results indicate that a form of computer-aided knowledge acquisition as instantiated by the Smalltalk-based KEATS system prototype is effective. A total of 29 hours of expert-system interaction (supervised by a mostly passive knowledge engineer) and 100 hours of protocol analysis produced new knowledge on two levels. Schemas for situation assessment and overall action plans were found on an upper or macro level. The existence of additional macro-level knowledge was inferred from differences in plans. On a lower or micro level, 64 rules for air request tasking were identified. Additional micro-level knowledge was identified in the form of requirements for functional changes of the ASOC facilities replicated in KEATS.

IV. DISCUSSION AND CONCLUSIONS

The premise for the work reported here is that instructional content for training decision-making skills can be acquired by means of knowledge engineering techniques.

Decision task knowledge is assumed to consist of more or less "fuzzy" rules-of-thumb, principles, or heuristics which make it possible for Simon's (1955) "rational man," with his limited information processing capabilities, to perform adequately and reliably in a complex, "messy" and time-constrained decision environment. It is this task knowledge that must be available in explicit form to the instructional designer to enable him to devise appropriate training. Manuals or other explicit documentation may contain isolated bits and pieces which are typically hard to separate from the primarily procedure-oriented content found in such sources. Traditional task analysis techniques, and even techniques specifically tailored to the elicitation of cognitive structures (Brecke et al., 1988), are ineffective means for externalizing expert knowledge deployed in complex cognitive tasks. Knowledge engineering methods, however, have proven to be effective for that purpose over a wide variety of subject-matter domains.

This effort employed knowledge engineering methods to acquire instructional content. Initially, a series of focused and structured interviews were held, taped, transcribed, and laboriously analyzed. The structured interviews revolved around hypothetical decision scenarios or cases which were presented on paper and maps. These "unaided" efforts were successful up to a point, although their inefficiency was a good demonstration of the knowledge acquisition "bottleneck." A broad surface layer of knowledge about the study domain was acquired, but attempts to achieve more depth were thwarted by the CATCH 22 problem. Some deep knowledge was required to develop elicitation stimuli which were rich and realistic enough to elicit more deep knowledge. The knowledge engineers did not have a critical mass of that deep knowledge to build into the paper-based cases. The latter were difficult to use since their rigid scripts made them very fragile. Continued access to the experts was also becoming a significant problem, which was at least partially dependent on their continuing motivation to support the project. An adequate QUID-PRO-QUO for the efforts of the experts was deemed to be useful in this regard.

KEATS was developed as a prototypical solution to these problems. It allows experts, who have the necessary deep knowledge, to design knowledge elicitation stimuli in the form of realistic decision scenarios ("exercises") for a given domain. Other experts can react to these scenarios by making decisions and by explaining the reasons behind their decisions. These reactions are captured by the system on a detailed protocol or transcript. The degrees of complexity and realism built into the scenarios are strictly a function of the objectives the scenario builder has in mind. KEATS supports training and mission

rehearsal objectives besides knowledge engineering objectives and thus offers a useful quid-pro-quo to supporting agencies in return for their experts' time. The system should also alleviate the access in that it runs on microcomputer hardware available to the experts at their peacetime duty stations and is equipped with a user interface which is designed for unassisted use by computer-naive personnel.

The question is now whether the technology represented by the KEATS prototype is useful and cost effective on a broader scale; for example, in other Tactical Command and Control nodes. The answer to that question depends on how well the prototype worked within the study domain and on the feasibility and cost of transferring the prototype technology to other domains. At this point in the project, there is only enough information for a somewhat speculative analysis and projection on both counts.

Efficacy of the Prototype

Only the KNOWLEDGE ENGINEERING function of KEATS has been subjected to limited testing. The TRAINING and BUILDING functions have not been tested by SMEs at all; neither have the REVIEWING and ADMINISTRATING functions. The view into the issue of prototype efficacy is therefore very narrow.

KEATS AS A PROTOTYPE SUPPORT TOOL FOR KNOWLEDGE ENGINEERING

The results of the first trial indicate that KEATS was effective as a support tool for KNOWLEDGE ENGINEERING. A solid and fairly large increment of new knowledge was obtained within 2 days. This knowledge product was the result of a number of factors intrinsic to KEATS and other extrinsic factors arising from constraints on the experts' time. Each of these factors is briefly discussed below:

MODE OF KEATS EMPLOYMENT

KEATS can be used either by an expert alone or by an expert and a knowledge engineer working together. During the first trial, KEATS was employed in the latter mode, which, as the results show, worked rather well. Whether and how well KEATS would work in the "expert alone" mode is still an open question. There is reason to believe that the output might have been less in quantity and quality if the knowledge engineer had not been there to prompt the expert with clarifying questions and to induce the expert to type in reasonings that he had uttered orally. On the other hand, an expert working alone might be less distracted and feel less inhibited and thus produce more natural output.

KEATS Fidelity

The functional fidelity provided by the KEATS environment was not perfect. A set of change requirements relating to functional fidelity was identified. These fidelity distortions are expected to have a

distorting effect on the knowledge obtained, especially if the expert cannot implement decision options he would use if there were better correspondence between the real world and the KEATS environment.

The potential knowledge contamination effect introduced by fidelity deficiencies is minimized if the expert indicates where the system forced him into a decision that he would not have made in reality, and if knowledge is validated over the course of repeated knowledge engineering sessions. The probability that isolated fidelity distortions will produce a systematic, wholesale distortion effect rather than isolated, occasional effects is low. Isolated distortions are identifiable because they are inconsistent with the rest of the knowledge base.

The precise and inspectable context presented by KEATS makes fidelity breaches very obvious. Notes regarding such breaches can be entered into the system on the spot simply by accessing a Suggestion Box. The conveniences offered by the Smalltalk environment allow rapid and efficient implementation of changes.

The identification of fidelity shortfalls is another form of knowledge acquisition which leads to improvements of the functional fidelity of the KEATS environment. The first trial was only the second time that SMEs had worked with the system. It is assumed -- and hoped -- that future system versions will show a steeply decreasing need for changes and that eventually a "perfect" KEATS will emerge. [3]

During the first trial, all instances of fidelity distortion were explicitly identified and noted. These notes were taken into account during subsequent rule formulation. We therefore conclude that the knowledge product of the first trial contains few, if any, distortions.

Knowledge Elicitation Questions

At the present time, KEATS has a particular knowledge elicitation strategy and a particular set of questions built into the knowledge engineering mode. Other strategies and questions are certainly possible. Differences in effectiveness and efficiency would have to be determined experimentally. The current strategy and question set were effective in the knowledge-engineer-assisted employment mode.

Two exceptions to effectiveness were noted. The micro-level question which asks for threshold conditions that would lead to alternative decisions was too open-ended and did not produce the desired types of responses. It is not clear whether the question was badly worded or whether it was probing for knowledge where none existed; i.e., the question may simply have been irrelevant to the experts.

[3] It is quite conceivable that system perfection and knowledge base compilation will coincide, since the system can be perfected only when the domain is fully understood.

The second exception was noted on the macro level; i.e., during the *Orientation Phase*. Two different experts were presented with identical situations and came up with two vastly different *Plans of Action*. The differences in these plans were to some extent attributable to differences in situation assessment, but not entirely. It appeared that the plan differences were also caused by differences in the experts' fundamental ideas about the role and purpose of the ASOC and the basic strategies to achieve that purpose. KEATS did not have a built-in knowledge elicitation mechanism to probe for this type of knowledge.

The issue of expert differences is clearly very important for training design. If experts are serving as models and if these models produce widely diverging outputs in response to the same situational stimuli, then which is the model that students should emulate? Part of the answer to this question lies in an examination of the knowledge differences which can account for these differences in output. KEATS should therefore include queries which probe for this type of knowledge. The most effective method would probably consist of an initial system-presented query, followed by one or more questions generated by a knowledge engineer.

User Input Modality

The expert has to type his responses to knowledge engineering queries. Most experts are not very good at typing. The process is therefore tedious, slow, and somewhat annoying. The expert may have a tendency to minimize the effort, and type as little as he feels he can get away with, instead of providing a full and elaborate account of his reasoning. During the trial, the experts had to be reminded occasionally to type in all they said, which may be the reason why the tapes did not contain more relevant information than the KEATS transcripts.

Typing (i.e., expression in writing) is also much more subject to editing than is expression in speech. The typed output may therefore be even further removed from the covert cognitive activity that actually led to the decision. The tapes did not show that the oral output was substantively different from the written output. How well either type of output represented the cognitive decision-making activity which preceded it is unknown.

Basically, typing did not seem to be an obstacle to knowledge input during the trial. The process was fairly slow but the impression was that it kept pace with the tempo of the experts' deliberations. There is no compelling reason at this stage to change the input modality to recorded speech.

Protocol Mechanism

All post-trial analysis was done from the transcripts. Given the results that were achieved in terms of distilled, formulated knowledge, one can unequivocally say that the current protocol mechanism is effective.

On the other hand, there are a number of information items which could be gathered easily and added to the transcript. For example, all consultations of information resources (status boards, briefings, queries addressed to ASOC internal or external parties) could be recorded. Passage of real time could be captured by noting the time elapsed between the presentation of an event and the entry of a decision response. Items entered on the Notepad or in the Suggestion Box could be shown in the transcript in the exact sequential order in which they occurred during knowledge elicitation.

Adding these items would result in a richer and more detailed protocol of the decision-making process. Analysts could see directly what information sources were consulted and in which sequence, how much time the expert spent with each decision problem, what he considered important for future reference (Notepad), and exactly when and where difficulties with fidelity were encountered (Suggestion Box). The additional information may make it easier to reconstruct the expert's decision path and therefore aid in interpretation of his responses to knowledge elicitation and in the formulation of rules.

The enriched protocol would also aid in getting answers to follow-on questions. A more detailed transcript (which can be inspected online in the REVIEWING mode) would make it easier for the expert to reconstruct his train of thought and to supply additional details. It should be noted that this follow-on questioning can be done remotely. All that is required is that the analyst and the expert have the same exercise run loaded into their respective computers. The analyst can enter the expert's added comments and explanations into the transcript on the spot.

Finally, enriched protocols may be of use to other researchers who are interested in unravelling basic issues regarding the cognitive processes which occur during decision making.

Current project plans include adding more information to the transcript.

Exercises Used

The exercises used during the first trial were made up by the knowledge engineers, rather than by the experts. The trial, therefore, did not provide a test of the assumption that expert-generated knowledge elicitation stimuli facilitate the acquisition of deep knowledge. It did provide, however, a benchmark reference against which future expert-generated exercises can be compared. The question as to whether KEATS is an effective solution to the CATCH 22 problem therefore remains open.

The acquisition of deep knowledge requires more than realistic stimuli. It also requires repeated, answer-dependent questions probing successively deeper layers of knowledge; i.e., a dynamic dialog between expert and inquisitor. Although there is some promise in current work on automated knowledge acquisition (Parsaye, 1988), practical application in a messy domain such as Tactical Command and Control remains far in the future. KEATS does, however, support and facilitate a dynamic dialog between a human knowledge engineer and the expert. It provides a very precise, unambiguous, inspectable, and realistic context within which effective and efficient communication between the expert and the knowledge engineer can take place. Our experience with both paper-based and machine-based (i.e., KEATS-based) cases indicates that the computer-based environment keeps knowledge elicitation much more focused and that it reduces, if not eliminates, unproductive excursions in the dialog. However, since the trial was not based on expert-generated exercises, it remains to be seen whether increased realism in the knowledge elicitation stimuli would make the expert-inquisitor dialog more productive in terms of depth of acquired knowledge.

Unknown at this point also is whether expert-generated exercises would indeed be much different from the knowledge-engineer-generated exercises that were used in the first trial. Our subjective impressions are that experts should have no problem whatsoever in developing more realistic exercises than we did. Experts should also have little problem with the mechanical aspects of exercise development. The BUILDING mode works flawlessly and the GUIDED submode should be very easy to master even for computer-naive personnel. Experts may have a problem in making available the time it takes to develop an exercise, but the BUILDING mode is probably the fastest tool currently available for constructing a complete and realistic exercise scenario.

Current project plans call for additional knowledge engineering sessions using expert-generated exercises.

Time

The experts could only make 2 days available for actual knowledge engineering (2 more days were used for travel). About 7 hours were spent on each exercise (one exercise per day). During these 7 hours, the experts completed, on the average, 73 minutes of exercise (or game) time. Each exercise covered 8 hours (i.e., the time period from early morning to late afternoon of a fictitious day in a war). This means that only about 15% of the *Operations Phase* of each exercise was completed; or, expressed differently, that time was slowed down by a factor of 7 during knowledge engineering. The *Orientation Phase* consumed approximately 2 of the 7 hours spent on each exercise.

As a consequence, no knowledge was obtained that would elucidate how decisions change as function of remaining daylight hours and as a function of depletion of aircraft resources. The scripts also contained a number of specific events that were to occur around noon and in the early afternoon, which were designed to access particular types of knowledge.

The trial, therefore, did not even begin to fully exploit the existing exercises. Complete knowledge engineering runs for each of the existing exercises would undoubtedly yield much more knowledge and thus provide a greater return on the investment of exercise preparation time. Given the fact that existing exercises can be edited into completely different exercises for a fraction of the effort involved in building an exercise from scratch, even higher cost/benefit ratios can be achieved.

Summary

The overall result of the first serious knowledge engineering application with KEATS is essentially a proof of concept. Clearly, the system has not yet reached maturity, which is not surprising given the complexity of the study domain. The overall design of the system as a knowledge engineering support tool appears to be sound. The requirements for functional improvements that were brought to light by the first trial suggest improvements of details rather than fundamental changes of concept. The detail changes are all technically feasible and, due to the efficient fast prototyping environment, easy and economical to implement.

The key issue that remains open is the effectiveness of KEATS as an instrument for breaking through the depth barrier; i.e., in getting around the CATCH 22 problem.

KEATS AS A PROTOTYPE TRAINING SYSTEM

The ultimate goal of this project is a technology for training decision-making skills. This technology must be tangibly instantiated with training system prototypes. KEATS is, on the one hand, a system that aids in the acquisition of instructional content required for building training system prototypes, and it is also a training system prototype.

KEATS has not been subjected to a formative evaluation as a training system; i.e., data which could give clues on its efficacy as a training system are not available. However, the technical features and characteristics of KEATS are known and enable speculation on the subject.

Basically, KEATS is capable of presenting practice stimuli in the form of decision problems, of enabling realistic information search activities, and of accepting decision responses. KEATS is, however, deficient in the area of feedback. Taylor (1983), in a review of literature relevant to unaided decision making, noted very accurately that developers of decision-making training programs face two problems:

"providing for consequences of decisions" and "evaluating decision making performance." KEATS does both, but only to a limited extent.

KEATS provides for consequences of decisions by manipulating airplane availability and by adjusting the script. Airplanes which are enroute can be tasked only by diverting them. Attrition occurs as a function of threat. KEATS provides for realistic consequences with respect to the air side of the battle, but it does not provide for decision consequences in terms of the ground war. The learner's decisions will have no influence on whether a scripted advance or retreat of opposing ground units takes place or not. He will therefore never know whether his manipulations of the supply side had any effect on the demand side of the equation. The features of KEATS which provide for consequences on the air side contribute much to the knowledge engineering function by making cases more robust (i.e., impervious to deviations from the script). They are also required for training, but by themselves, they provide a skewed, one-sided picture.

KEATS also evaluates decision-making performance, but again, only to some extent. Nickerson and Feehrer (1975) were adamant in suggesting that decision-making performance should be evaluated in a partial *a priori* fashion; i.e., on the basis of how well the decision maker used available information at the time the decision was made. They felt that *a posteriori* evaluation (i.e., evaluation based on the effects of the implemented decision), especially in complex domains, could not be accomplished at all since the effects are inevitably confounded by factors not under the decision maker's control or cognizance. KEATS does perform a partial a priori evaluation of a decision. It checks whether the decision would violate any physical constraints and it does provide appropriate feedback. KEATS thus examines the technical feasibility of a decision, but it has nothing to say about its tactical wisdom. If KEATS is used with an instructor or coach, the latter can provide that missing aspect of feedback. In the instructor-assisted mode, KEATS *can* provide a full measure of a priori decision evaluation (if the instructor is qualified).

Thus, KEATS is not yet the training system that this project is aiming for (see discussion of project goals at the end of Section I). It does satisfy the requirement of running on low-cost hardware.[4] However, it still requires an instructor for decision-making training. If and when KEATS reaches full fidelity, it will be a fully functional procedures trainer in the stand-alone mode. Its adaptability to local needs, particular scenarios, and individual trainee requirements as afforded by the BUILDING mode makes it more flexible and economical than the paper-based System Training Exercises which are currently used by the ASOC and other TACS nodes. However, as a decision training system, KEATS is merely a beginning.

[4]Standard squadron-issue microcomputers need to be upgraded from 640K bytes to 4M bytes.

KEATS AS A WHOLE

Given the limited empirical data regarding KEATS' efficacy in either of its major functions, it is not possible to state unequivocally whether the problems that gave rise to its development have indeed been solved. KEATS does present robust cases for knowledge engineering and this robustness should increase with increasing functional fidelity. The environment it presents to the experts is already convincing enough to fully engage their operational decision-making skills. As stated before: There are no final answers on the CATCH 22 problem; however, there is good reason to believe that KEATS can be a facilitating factor in this regard. There are no final answers on the ACCESS and the QUID-PRO-QUO problems either. The system has been at the ASOC for only 3 months and during most of that time, ASOC personnel were out on exercises.

Transfer to Other Domains

The answer to the first question posed at the beginning of this discussion ("Does the prototype work in the study domain?") is tentative and incomplete. In view of that, it may be too early to deal with the issue of generalizability to other domains. However, the prototype concept was confirmed as basically sound, and the knowledge product of the first trial was substantial for a first formative trial. Some speculative extrapolation to other domains appears permissible and useful at this point.

What are such "other" domains? KEATS was designed for the ASOC domain, which was identified as a domain that is representative of Tactical Command and Control in the Air Force. There is no apparent reason why KEATS could not be expanded to include all the objects that the larger world of an Allied Tactical Operations Center (ATOC) includes. By the same token, there is no reason why the same technology that was used in KEATS could not be applied to a Wing Operations Center (WOC) or a Tactical Air Control Party (TACP).

The key to the issue of transferability is in the modeling requirements for decision problem presentation and for decision implementation. As long as decision input and output for a domain are in the form of verbal messages, KEATS technology is directly applicable. It would not be directly applicable, for example, to the decision problems in an airplane or in a nuclear power plant. In those instances, decision input (problem presentation) is in the form of complex visual images and dial indications. Decision output is in the form of direct control manipulations. The control manipulations must be followed by more or less instantaneous feedback. A full simulation of the physical system is required in these cases. Verbal decision input and output are common across the domain of Air Force Tactical

Command and Control and therefore KEATS is at least transferrable to other Air Force Tactical Command and Control nodes. To the extent that the input/output conditions are satisfied, it is also directly applicable to other decision environments.

Direct transfer or direct applicability is understood quite literally. The KEATS prototype consists of domain- or application-specific code and generic code. The generic code (object classes) represents about 70% of KEATS, and it is that portion that can be directly transferred to another domain. The objects of the new domain must be added and their behavior coded as "methods." The latter is a non-trivial undertaking but the object-oriented fast prototyping environment offered by Smalltalk reduces labor requirements. The combination of reusable code and fast prototyping technology makes transfer to other domains an economically attractive undertaking.

Cost effectiveness of transfer may, however, be a moot point if there is no other way to acquire the knowledge. Our evidence suggests that knowledge acquisition for decision making in a "messy" domain could not be done effectively, if at all, with interview-type techniques. KEATS provides the necessary structure and precision for effective expert-knowledge engineer communication and prevents the type of chaos that can ensue from imprecisely communicated and understood questions and answers. More than that, a KEATS-type system grows in the course of its application as a knowledge engineering system into a precursor or prototype of the target training system. In this project, KEATS has provided a baseline for what the final training system should look like and do. Our general approach to this project has therefore become an evolutionary rather than a serial approach. It is interesting to note that a similar basic approach is currently being pursued by the Army Research Institute (Stoddard et al., 1986), where a first training system prototype is being used to assist in the acquisition of knowledge for a cognitive skills tutor.

V. FUTURE DIRECTIONS

By the end of the third and final project year, we hope to have implemented SuperKEATS, a prototype decision-making training system which goes beyond the existing KEATS in a number of aspects. SuperKEATS will represent the confluence of the three major project activities (see Figure 1): the acquisition of instructional content (which was the subject of this report), the search for instructional strategies, and the implementation of content and strategy within constraints imposed by existing computer technology.

The most significant difference between the two systems will be in the area of feedback. SuperKEATS will be capable of providing for decision consequences. The key to that capability is an underlying simulation of an Army Corps' ground battle and the effects of air support on that battle. It is hoped that scenario events will no longer be scripted but generated extemporaneously. The course of the battle will be determined by the relative strengths of the opposing ground forces and by the amount and effectiveness of air support supplied to friendly ground forces. The decision-making performance of the "ASOC student" will therefore have plausible real consequences, but consequences which are not completely determined by ASOC decisions alone (as in reality!).

SuperKEATS will hopefully also be capable of providing *a priori* decision evaluations which cover tactical appropriateness (or tactical wisdom), as well as technical feasibility. The prerequisite for that capability is a further increment of knowledge which we intend to acquire via KEATS. The near-term target in terms of knowledge acquisition is a set of criteria which make a particular tasking decision appropriate or inappropriate with respect to a given situation assessment and plan of action.

Given these expanded feedback capabilities, a similar user interface, a capability for constructing a starting scenario, and an instructional strategy module that can be adjusted, SuperKEATS will be a vehicle for decision-making training research which will allow investigation of a wide variety of issues related to the performance of decision-making tasks, the acquisition of decision-making skills, and the effectiveness of training strategies.

REFERENCES

Barnthouse, D.A. (1989). Tactical Air Operations Team Training System (TAOTTS) functional description (Unpublished manuscript).

Brecke, F.H., Jacobs, F.B., & Krebs, J. (1988). Improved training of battlestaff and commanders assigned to tactical command and control (C^2) systems (AFHRL-TP-87-38, AD-B123-229L). Wright-Patterson AFB, OH: Logistics and Human Factors Division, Air Force Human Resource Laboratory.

Frank, H.G. (1969). Kybernetische Grundlagen der Paedagogik (rev. ed.). Baden-Baden, Germany: Agis-Verlag.

Fraser, B.D. (1987). Knowledge acquisition methodology (Technical Report 1094). San Diego, CA: Naval Ocean Systems Center.

Hayes-Roth, F., & Waterman, D.A. (1984). An investigation of tools for building expert systems. In F. Hayes-Roth & D.A. Waterman (Eds.), Building Expert Systems. Reading, MA: Addison-Wesley Publishing Company.

Johnson, P.E. (1983). What kind of expert should a system be? The Journal of Medicine and Philosophy, 8, 77-97.

Madni, A.M., Ahlers, R., & Chu, Y. (1987, December). Knowledge-based simulation: An approach to intelligent opponent modeling for training tactical decisionmaking. In Proceedings for the Ninth Interservice/Industry Training Systems Conference. Washington, DC: National Security Industrial Association.

McCune, B.P. (1985). A tutorial on expert systems for battlefield applications. In Proceedings of the Seminar on Artificial Intelligence. Applications to the Battlefield. Ft. Monmouth, NJ: Armed Forces Communications and Electronics Association.

Merrill, M.D., & Wood, N.D. (1974, April). Instructional strategies: A preliminary taxonomy. Paper presented at the meeting of the Special Interest Group for Research in Mathematics Education, American Educational Research Association, Chicago, IL.

Montague, W.E. (1986, April). Application of cognitive science principles: Instructional heuristics and mechanisms for use. Paper presented at the annual meeting of the American Educational Research Association, San Francisco, CA.

Nickerson, R.S., & Feehrer, C.E. (1975). Decision-making and training: A review of theoretical and empirical studies of decision-making and their implications for the training of decision makers (NAVTRAEQUIPCEN 73-C-0128-1). Cambridge, MA: Bolt Beranek and Newman, Inc.

Obermayer, R.W., Johnston, D.L., Slemon, G.S., & Hicklin, M.B. (1984). Team training: Knowledge-based simulation for team members (NAVTRAEQUIPCEN 82C-0140-1). Orlando, FL: Naval Training Equipment Center.

Parsaye, K. (1988). Acquiring & verifying knowledge automatically. AI Expert, 48-63.

Priest, P.F. (1986, May). DARPA's AirLand Battle Management Program and USAF's Tactical Expert Mission PLAnneR (TEMPLAR). In Advanced Computer Aids in the Planning and Execution of Air Warfare and Ground Strike Operations: Conference Proceedings (AD-A182 096), Meeting of the Avionics Panels of AGARD (51st), Kongsberg, Norway.

Rasmussen, J. (1986). Information processing and human-machine interaction: An approach to cognitive engineering (Series Volume 12). New York: NorthHolland.

Schraagen, J.M.C. (1986, December). Expert differences and their implication for knowledge elicitation techniques (Report IZF 1986-34). Kampweg 5, The Netherlands: Institute for Perception.

Simon, H.A. (1955). A behavioral model of rational choice. Quarterly Journal of Economics, 69, 99-118.

Stoddard, M.L., Kern, R., & Emerson, J. (1986, March). A computer-based knowledge extraction tool: A step in the development of a cognitive skills tutor. Submitted to the Association for the Development of Computer-based Instructional Systems. Alexandria, VA: U.S. Army Research Institute for Behavioral and Social Sciences.

Taylor, E.N. (1983). A review of literature relevant to unaided tactical decision-making (Research Note 83-35). Alexandria, VA: U.S. Army Research Institute for Behavioral and Social Sciences.

Waterman, D.A. (1986). A guide to expert systems. Reading, MA: Addison Wesley Publishing Company.

Wickens, C.D. (1984). Engineering psychology and human performance (Decision-Making, Chapter 3). Columbus, OH: Merrill Publishing Company.

Wilson, J.O. (1982). Interactive microcomputer wargame for air battle (Master's thesis). Monterey, CA: Naval Postgraduate School.

ABBREVIATIONS AND ACRONYMS

ALO	Air Liaison Officer
ARTACT	Armor Tactical Concepts Tutor
ASOC	Air Support Operations Center
ASW	Anti-Submarine Warfare
ATO	Air Tasking Order
ATOC	Allied Tactical Operations Center
C^2	Command and Control
CAI	Computer-Aided Instruction
CPX	Command Post Exercise
DTG	Date-Time-Group
EIDS	Electronic Information Delivery System
FDO	Fighter Duty Officer
FTX	Field Training Exercise
ISA	Initial Situation Assessment
ISD	Instructional Systems Development
KEATS	Knowledge Engineering and Training System
MIPS	Millions of Instructions Per Second
NCO	Noncommissioned Officer
OAS	Offensive Air Support
OO	Operations Order
POA	Plan of Action
SME	Subject-Matter Expert
TACC	Tactical Air Control Center
TACP	Tactical Air Control Party
TACS	Tactical Air Control System
TAOTTS	Tactical Air Operations Team Training System
TEMPLAR	Tactical Expert Mission Planner
TOT	Time Over Target
UCSD	University of California at San Diego
WOC	Wing Operations Center

APPENDIX

KEATS OBJECT CLASSES

S **Behavior**
S Class
S MetaClass
S **BitBlt**
S CharacterScanner
S Pen
S Animation
S Commander
S **Boolean**
S False
S True
C **CC**
C CCentry
C **CCbrowser**
S **ClassBrowser**
S **ClassHierarchyBrowser**
S **ClassReader**
S **Collection**
S Bag
S IndexedCollection
S FixedSizeCollection
S Array
S CompiledMethod
S Bitmap
S ByteArray
S FileHandle
S Interval
S String
S Symbol
S OrderedCollection
S Process
S SortedCollection
S Set
S Dictionary
S IdentityDictionary
S MethodDictionary
S SystemDictionary
S SymbolSet
S **Compiler**
S LCompiler
S **Context**
S HomeContext
S **CursorManager**
S NoMouseCursor
S **DeletedClass**
S **DemoClass**
S **Directory**
S **DiskBrowser**
S **Dispatcher**
S PointDispatcher
S ScreenDispatcher
S ScrollDispatcher
S GraphDispatcher
C ActiveImageDispatcher
S ListSelector
C ReadOnlyDispatcher
S TextEditor
C InformEditor
C PopUpEditor
C NoScrollEditor
C SingleLineEditor
C PromptEditor
S TopDispatcher
S **DispatchManager**
S **DisplayObject**
S DisplayMedium
S Form
S BiColorForm
S ColorForm
S DisplayScreen
S ColorScreen
S **Dos**
C **Dummy**
C **Editor**
M Environment
Board
IntelBoard
IntelLine
MissionLine
StatusBoard
TacpBoard

LEGEND:

S = Smalltalk
C = KEATS Core, Transportable As Is
M = KEATS Core, Transportable with Minor Changes
No Code = Application Dependent

TacpLine
EnvironmentEvent
HistoricalEvent
EnvironmentType
AircraftType
EwEcmAcType
FacAcType
FighterAcType
DivisionType
OrdnanceType
WxType
BaseWxType
GeneralWxType
M ExerciseHeader
GeneralBackground
Geography
History
RulesAndAssumptions
PhysicalObject
Aircraft
Base
ForceUnit
AirCommandControl
Asoc
AtocOrTacc
Tacp
Woc
AirUnit
F-
Squadron
GroundUnit
Corps
EnemyCorps
FriendlyCorps
Division
EnemyDivision
FriendlyDivision
GroundAirThreat
EnemyThreat
FriendlyThreat
Nature
GeneralWx
CurrentGeneralWx
ForecastGeneralWx
Sun
Person
Alo
Arno
AsocPerson
AtocWocPerson
AtocPerson
WocPerson
Director
Fac
FighterDutyOfficer
Fido1
Fido2
G3Air
Intel
LogiconPerson
OtherAsoc
OtherPerson
Recce
TacpPerson
SystemObject
AtmInProcess
CandidateAircraft
WOCResponseMessage
TacticalObject
Briefing
CommunicationsBriefing
DirectorBriefing
DirectorWrapUp
FDOBriefing
FDOShiftChangeBriefing
G3AirBriefing
IntelBriefing
MaintenanceBriefing
RecceBriefing
RecceShiftChangeBriefing
WeatherBriefing
Guidance
AirForceGuidance
ArmyGuidance
Line
Flot
Fscl
Ripl
Mission
OrderOfBattle
AirOrderOfBattle
EnemyAOB
FriendlyAOB
EnemyOrderOfBattle
GroundOrderOfBattle
EnemyGOB
FriendlyGOB
Orders
AirCoordinationOrders
AirTaskingOrders
DailyOperationsOrders
TacticalSituation

LEGEND:

S = Smalltalk
C = KEATS Core, Transportable As Is
M = KEATS Core, Transportable with Minor Changes
No Code = Application Dependent

EnemyTacticalSituation
FriendlyTacticalSituation
M Event
ResponseEvent
AtrResponseEvent
AtrResponse
DeleteAtrResponseData
MakeAtrResponseData
CommentEvent
UserComment
Assessment
OtherComment
PlanOfAction
Rationale
TellLogicon
NoChangeActionEvent
AskEvent
FactLookupEvent
LookAtBriefingsEvent
LookAtOrdersEvent
LookAtStatusBoardEvent
StatusBoardEvent
DeleteLineEvent
DeleteMissionLineEvent
DeleteIntelLineEvent
DeleteTacpEvent
InsertMissionLineEvent
InsertIntelLineEvent
InsertTacpEvent
IntelHeaderEvent
MakeDuplicateMissionLine
MakeMissionLineEvent
MakeIntelLineEvent
MakeTacpEvent
MakeStatusBoardEvent
MissionLineEvent
IntelLineEvent
TacpEvent
MoveMissionLinesEvent
StatusBoardAirUnitsEvent
StatusBoardBaseEvent
StatusBoardCommentEvent
StatusBoardDivisionEvent
StatusBoardHeaderEvent
TellEvent
AtocWocTell
EstablishSmoothFlow
RecommendFlush
RecommendRelocate
TellHoldAircraft
TellPutOnAlert
TellRemoveFromAlert
G3AirTell
ImpendingAirAssetDepletion
RecommendForgetPercentages
UnableToMeetRequests
PersonTell
TellOther
ScenarioEvent
AirUnitEvent
FlushAirUnit
GainFighterAircraft
LoseFighterAircraft
MakeAirUnitAvailable
RelocateAirUnit
AtrEvent
WocRefuseAtr
AtrResponseData
BaseEvent
CloseBase
OpenBase
CheckAtrEventProcessing
DivisionChange
DivisionMovement
EnemyThreatChange
FirePriorityChange
FlotUpdate
G3AirUpdate
GuidanceChange
IntelUpdate
MakeMissionAvailable
WxEvent
BaseWxChange
GeneralWxChange
ForecastWxChange
M SystemEvent
M EndEvent
M StartEvent
M ParameterDescription
PrintAtr
Thesaurus
S **EmptySlot**
S **File**
S **Font**
S **ForwardReference**
S **Icon**
S **InputEvent**

LEGEND:

S = Smalltalk
C = KEATS Core, Transportable As Is
M = KEATS Core, Transportable with Minor Changes
No Code = Application Dependent

S **Inspector**
S Debugger
S DictionaryInspector

M **KEATS**
M KEATSinterface
M KEATS windowBrowser
C AdministrationBrowser
C AdminFunctionBrowser
C ConfigFunctionBrowser
C EditExerciseDescriptionBrowser
C PrivateExerciseBrowser
C PublicExerciseBrowser
C UserProfileBrowser
M BuildBrowser
M BuildExerciseDescription
M BuildEditExerciseDescription
M BuildReadExerciseDescription
M BuildExerciseSelection
M BuildFullDescription
M BuildOptions
M BuildSelectFactTopic
M BuildSelectScenarioTopic
M BuildVerification
M FixScenario
M FixScript
M GuidedBuild
M GuidedFactManipulation
M GuidedAllFacts
M GuidedScenarioManipulation
M GuidedAllScenario
M GuidedScriptManipulation
M UnguidedBuild
M UnguidedFactManipulation
M UnguidedScenarioManipulation
M UnguidedScriptManipulation
M ExerciseDescriptionBrowser
M GeneralHelpBrowser
KEATSRequestBrowser
KEATSResponseBackground
KEATSResponseBrowser
AlertResponseBrowser
GlowResponseBrowser
RemoveAlertResponseBrowser
M KnowledgeEngineeringBrowser
M ActionsBrowser
M AssessmentPlanBrowser
M CommentsBrowser
M ConditionsBrowser
M FactorsBrowser
M OptionsBrowser
TaskingOptionBrowser
M VarInConditionsBrowser
C ModeSelectionBrowser
C ReviewBrowser
C ReviewRunSelectionBrowser
C ReviewRunTextBrowser
C ReviewUserSelectionBrowser
C SuggBoxHelpBrowser
C SuggestionBoxBrowser
C TrainingBrowser
M AskTellBrowser
ASOCvanBrowser
ATRBottomsListBrowser
ATRDeleteBottomBrowser
ATRListBrowser
BriefingBrowser
BriefingsListBrowser
M FactLookupBrowser
C HelpBrowser
IntelBrdBrowser
M MessageDeskBrowser
C NotepadBrowser
OrdersBrowser
OrdersListBrowser
PastTaskingsBrowser
TacpBrdBrowser
TaskedATRsBrowser
TaskingFunctionBrowser
TrainingStatBrdBrowser
BuildStatBrdBrowser
TrainingStatBrdSelBrowser
BuildStatBrdSelBrowser
UntaskedATRsBrowser
C TrainingExerciseSelection
C UserNameSelectionBrowser
C WelcomeBrowser
C Parameter
C ListParameter
C HotSpotsParameter
C SelectorDescription
C KEATSUser
C KEATSutilities
C Formatter
C PopUps
C Advisor
C ErrorAdvisor
C MenuAdvisor
C TimeLimitInformMenu
C KEATSmenu
C ExtendedMenu
C ConfirmMenu

LEGEND:

S = Smalltalk
C = KEATS Core, Transportable As Is
M = KEATS Core, Transportable with Minor Changes
No Code = Application Dependent

C ListEntryMenu
C OptionsMenu
C YesNoMenu
C KEATSprompter
C MultiLineEntryPrompter
C LargeEntryPrompter
C SingleLineEntryPrompter

S **Loader**

S **Magnitude**
S Association
S Character
S Date
S Number
S Float
S Fraction
S FixedNumber
S Integer
S LargeNegativeInteger
S LargePositiveInteger
S SmallInteger
S Time

C **Manager**
C AgendaManager
C Case
C CaseLibrarian
C CaseVerifier
C DateTime

S **Menu**

S **Message**

S **MethodBrowser**

S **Pane**
S SubPane
C ExtendedListPane
C ExtendedColumnPane
S GraphPane
C ActiveImagePane
S ListPane
C ButtonPane
C StringListPane
C TempELP
S TextPane
C FillInTheBlanks
C InformPane
C ColumnInformPane
C NoScrollTextPane
C SelectableInformPane
C WPpane
S TopPane
C KEATSTopPane

S **Pattern**
S WildPattern

S **Point**

C **Printer**
C IBMgraphics
C LaserJet

S **ProcessScheduler**

S **Prompter**

S **Rectangle**

S **Semaphore**

S **Stream**
S ReadStream
S WriteStream
S ReadWriteStream
S FileStream
S TerminalStream

S **StringModel**
C WPmodel
C WP1model

C **Switch**

S **TextSelection**

S **UndefinedObject**

LEGEND:

S = Smalltalk
C = KEATS Core, Transportable As Is
M = KEATS Core, Transportable with Minor Changes
No Code = Application Dependent

☆ U.S. GOVERNMENT PRINTING OFFICE: 1989—661-041/9-00060